# Outils de multi-casting audio / vidéo / données avec PureData

# pour Access Grid

# MARCEL Multi-casting Platform

*MAJ 11.02.13*

**Le but de ce document** est de **proposer une vision de la meilleure façon d’intégrer pure-data** (pd) (<http://puredata.info/>) **dans access grid** (<http://www.accessgrid.org/>).

Cette intégration se ferait selon les préoccupations du réseau MARCEL. Ce document fait principalement l'inventaire des solutions existantes déjà intégrées à pd, et propose une architecture logicielle permettant d'atteindre au mieux les buts du projet par rapport aux contraintes matérielles.

L’intégration de pure-data dans access grid passe par l’implémentation d'outils permettant le multicast audio / vidéo et de données. Cette implémentation doit respecter les contraintes suivantes :

* cross plate-forme
* open source
* nombreux paramètres de configuration des systèmes afin de permettre une meilleure utilisation de la bande passante pour optimiser à la fois la qualité et la latence.

Le fait de posséder les outils de ce type dans pure data n'est que la première étape à l’intégration de pd dans access-grid. Une 2eme étape consisterait à créer des patch pd et les lancer depuis access grid avec les bons paramètres afin de les configurer automatiquement (adresse de multicast / réglage du débit et de la latence). Cette intégration permettrait dans une 3eme étape l'ajout de fonctionnalités qui ne sont pas supporté actuellement :

* sélection de l'affichage
* effets vidéo
* effets audio (spatialisation / compression)
* enregistrement des données

Le fait d’insérer pd dans la chaîne ne résout malheureusement pas tous les problèmes. Notamment :

* très faible latence
* distribution sur le web
* automatic dynamic remixing
* firewall et ouverture de ports

Les flux vidéo / audio et de données sont généralement traités de façon indépendantes. Nous allons voir ces aspects séparément.

## Flux vidéo

Choix de la libraire de traitement vidéo dans pd

Plusieurs libraires proposent l'ajout de fonctionnalités vidéo dans pure data

|  |  |  |
| --- | --- | --- |
|  | Plate-forme supportées | Développement actif |
| Freimstein | Win | non |
| pdp | OsX, Linux | non |
| gridflow | Win, osX, Linux | non |
| Gem | Win, osX, Linux, autres | oui |

<http://gridflow.ca/>

<http://zwizwa.be/pdp/>

[http://gem.ie m.at/](http://gem.iem.at/)

Gem est la seule librairie dont le développement soit encore actif. De plus, elle est cross plate-forme.

Son architecture est modulaire, permettant ainsi d'ajouter des Plug-in de réception ou d’émission de flux.

Il semble donc naturel de se concentrer sur cette librairie.

Limitation hardware

GPU (central processing unit)

Toutes les opérations génériques dans l'ordinateur transitent par le CPU. C'est lui qui va traiter les informations provenant de la carte réseau, et communiquer avec le GPU.

GPU (Graphical processing unit)

C'est la partie dédiée à l'affichage. Grâce à la demande du jeu vidéo, les GPU sont devenu très puissants. Sur un ordinateur moyen actuellement, le GPU est plusieurs centaines de fois plus performant que le CPU. C'est lui qui effectue les calculs graphiques complexes : déformation / couleurs, ainsi que la synthèse d'image (openGL). C'est le GPU qui envoie les informations sur les sorties vidéo de l'ordinateur.



Malheureusement, l'architecture matérielle est nettement asymétrique : l'envoie du CPU vers le GPU est énormément plus rapide que du GPU vers le CPU. Il est donc facile d'envoyer une image (un flux) en haute qualité au GPU, de la traiter localement afin de l'afficher. Par contre, il n'est pas possible de rapatrier cette image après traitement du GPU vers le CPU afin d’être compressée pour être envoyée sur le réseau. Il n'est pas non plus possible de traiter l'image dans le CPU. Le besoin d'une communication rapide GPU vers CPU étant marginal, les performances ne se sont pas améliorées de façon significative ces dernières années. Il n'est pas judicieux de miser sur une amélioration future de ce coté.

Par exemple : sur un ordinateur portable possédant une carte graphique Geforce GT 425M (moyen de gamme), passer une image full HD 1920x1080 pixel prend environs 100ms soit une fréquence maximum de 10 fps.

Envoi de flux vidéo après traitement par pd

Des solutions ne nécessitant pas de matériel spécifique existent, mais se heurtent au problème matériel décrit précédemment. Il est en effet possible de récupérer l'image générée par le GPU dans le CPU au prix des performances. Une fois dans le CPU, cette image peut être utilisée à la place d'images provenant d'une webcam. Comme par exemple avec videoloopback sous Linux.

<http://www.lavrsen.dk/foswiki/bin/view/Motion/LoopbackDevice>

Même si cette solution permet d'utiliser les outils actuels fonctionnant avec une webcam, elle n'offre pas de bonnes performances et n'est pas détaillée ici.

Par contre, une fois que l'image générée par pd est dans le CPU, les outils génériques sont utilisables. Même si le problème du passage GPU vers CPU ne rend pas viable cette solution.

Il est cependant possible d'utiliser la sortie vidéo pour générer un flux vidéo de bonne qualité. Cette solution consiste à utiliser une carte d’acquisition vidéo permettant ainsi de rapatrier l'image générée par le GPU dans le CPU. Cette carte d’acquisition branchée sur la sortie écran peut être branchée soit dans l'ordinateur générant le contenu vidéo, soit dans un autre ordinateur.



Finalement, cette solution consiste à remplacer une webcam par une carte acquisition vidéo. Elle est donc tout à fait envisageable avec des outils génériques, notamment avec les outils actuels.

Envoi de flux vidéo non traités depuis pd

Une camera (ou une carte d’acquisition vidéo) envoie son flux vidéo au CPU. Tant que ce flux n'est pas envoyé au GPU pour être traité, il reste disponible dans le CPU et peu donc être utilisé afin d’être envoyé sur le réseau. La solution la plus intéressante pour cela consiste à utiliser la librairie gstreamer :

« GStreamer is a library for constructing graphs of media-handling components. The applications it supports range from simple Ogg/Vorbis playback, audio/video streaming to complex audio (mixing) and video (non-linear editing) processing. »

« GStreamer has been ported to a wide range of operating systems, processors and compilers. These include but are not limited to Linux on x86, PPC and ARM using GCC. Solaris on x86 and SPARC using both GCC and Forte, MacOSX, Microsoft Windows using MS Visual Developer, IBM OS/400 and Symbian OS. » »

<http://gstreamer.freedesktop.org/>

Le projet scenic de la SAT utilise ce toolkit pour la partie streaming / multicasting, à la fois pour l'audio et la vidéo. Scenic apporte notamment une interface graphique à gstreamer.

<http://code.sat.qc.ca/redmine/projects/scenic/wiki/Scenic_fr>

Mais il est aussi possible d'utiliser gstreamer depuis pd grâce à la librairie pdgst :

<http://umlaeute.mur.at/Members/zmoelnig/projects/pdgst>

Cette libraire n'est malheureusement pas entièrement aboutie, et le développement n'est pas actif, mais il semble tout a fait possible de le poursuivre.

Réception de flux vidéo depuis Pure Data

En dehors de gstreamer qui propose à la fois l'envoie et la réception, Gem propose un plug-in permettant d'utiliser VLC (Video Lan Client) comme librairie de réception vidéo.

<http://www.videolan.org/vlc/features.html>

VLC est un logiciel libre, fonctionnant sur de nombreuse plat forme. Il est compatible avec de nombreux type de flux, compatible avec le multicast (C'est le logiciel recommandé par Free pour recevoir la télévision sur un ordinateur). Le noyaux de VLC est développé comme une librairie qui offre de nombreux paramètres.

Ce plug-in permet d'utiliser pd pour recevoir et donc traiter et afficher des flux provenant de la majorité des serveurs de stream vidéo. Il est donc relativement facile d'utiliser pd pour recevoir des flux.

Latence vidéo

Une webcam standard affichée localement (donc sans buffer spécifique), dispose d'une latence totale d'environ 200ms. En utilisant une camera rapide (60fps), et divers optimisations dans pd (notamment à l'affichage), il a été possible de baisser la latence d'affichage à 70ms.

L'envoie d'un flux d'image à travers un réseau augmente considérablement cette latence, quelque soit la vitesse de ce réseau. Ces données sont utilespour garder un objectif réaliste sur la latence vidéo possible.

## Flux audio

Il existe plusieurs solutions actuellement permettant l'envoi de sons à travers un réseau depuis pd.

udpsend~ / udpreceive~

Ces objets permettent l'envoi de son non compressé. Voici l'aide de ces objets :



Le multicast est implémenté. Ainsi que diverses options de qualité et de bufferring du flux. Cependant le débit utilisé par de l'audio non compressé est important, il est en général recommandé d'utiliser un flux audio compressé.

Oggcast~ / oggamp~

Ces objets permettent l'envoi et la réception de son compressé via un serveur Jroar ou Icecast2. Ces objets sont configurables à la fois au niveau de la qualité audio ainsi que de la taille de buffers.

Voici l'aide de ces objets :



Cette solution impose l'envoie du son au serveur, avant d’être partagé sur le réseau. Le serveur peut être installé localement ce qui semble plus laborieux mais permet de réduire la latence. A l'inverse, un serveur unique peut être utilisé, mais la latence risque d’être plus élevé.

Le serveur icecast ne fonctionne que sous linux/unix et windows.

Le server Jroar, en java (donc cross plates-formes) semble plus simple à déployer.

Cette solution n'est donc pas la plus simple à mettre en œuvre, puisqu'elle nécessite l'installation (et la configuration) d'un logiciel extérieur. Par ailleurs, ce système n'est pas robuste en cas de perte de paquet, ou simplement si le buffer de réception est vide. En pratique, ce système a été développé pour des applications de web radio et n'a pas été optimisé pour obtenir de faibles latences. Il semble difficile d'obtenir des latences de moins de 10s ce qui n'est pas acceptable pour cette application.

Gstreamer

Gstreamer permet de traiter l'audio aussi bien que la vidéo. Les 2 peuvent même être multiplexés ensemble afin de garder leur synchronisation lors de la transmission. Il semble que ce soit la seule solution permettant d'atteindre les objectifs fixés.

## Flux de données

Plusieurs objets dans pd proposent l'envoi de flux de données de contrôle.

netsend / netreceive

Ce sont les objets natifs de pd, il permettent de faire transiter des données ascii en TCP ou en UDP. Ils ne sont pas compatibles multicast, et l'ascii semble un peu trop limitant pour ces applications.

tcpsend / tcpreceive / tcpclient / tcpserver

Ces objets proviennent de la librairie mrpeach, inclus dans pd-extended. Ils permettent l'envoi de données en TCP, ce qui ne permet pas l'envoi en multicast.

udpsend / udpreceive

Ces objets proviennent de la librairie mrpeach, inclue dans pd-extended. Ils permettent l'envoi de données binaire en UDP. Le multicast est supporté. Il semble donc que ces objets sont directement utilisables pour l'envoi de données dans notre cadre d'utilisation.

## Architecture proposée :

Nous avons vu que la vidéo ne peut pas être traitée avant d’être envoyée, on peut par contre envisager de la traiter localement, après réception du flux. Pour cela, il est nécessaire d'envoyer aussi les données de contrôle des effets.

Selon cette architecture, les données de contrôle sont utilisées pour piloter les traitements vidéos effectués sur les ordinateurs avant visualisation. Cependant, il est aussi nécessaire de permettre un contrôle et donc une visualisation avant l'envoi sur le réseau, ce qui donnerait :



Conclusion

Des limitations hardware qui n’évoluerons pas significativement empêche l'envoi de vidéo après traitement local. Il faut donc : soit utiliser une carte d’acquisition vidéo fonctionnant ainsi quelque soit la source vidéo, soit effectuer le traitement vidéo localement, ce qui complexifie l'architecture, et nécessite une puissance de calcul correspondant au traitement désiré sur les ordinateurs recevant le flux vidéo.

Il n'a été exploré ici que les solutions déjà implémentées dans pd. Parmi toutes ces solutions, l'utilisation du framework gstreamer semble le plus prometteur pour l'envoi de son / images.

Le travail consisterait donc en différentes phases :

- améliorer l’intégration de gstreamer dans pd ainsi que sa distribution

- utiliser gstreamer comme plat-forme de multicasting dans pd

- intégrer ces patchs de façon à pouvoir les lancer depuis accessgrid

Finalement, afin de satisfaire aux attentes en terme de latence / qualité d'images / etc, il sera probablement nécessaire d’apporter des améliorations à gstreamer. Ces développements devront se faire en bonne entente avec la communauté des développeurs de gstreamer.

L'envoi de données de contrôles peut se faire à partir des objets udpsend / udpreceive. Aucun développement supplémentairement n'est à prévoir pour cette partie. Cela peut donc constituer un point de départ pour l’intégration de pd dans access grid.

Il semble cependant important de noter : Pd est un environnement de programmation : même si il est possible de faire des patchs génériques intégrés à access grid, il faudra être un utilisateur de pd pour les modifier et les adapter / optimiser aux besoin du projet. Pd ne doit pas remplacer les librairie par défaut pour l'envoie et la réception audio et vidéo.

Cyrille Henry, Paris, 12/2012

Pour :

Aristote

MARCEL

Slider Lab, EESI

L'ange Carasuelo

Contact :

Don Foresta <don@donforesta.net>