Alexandre Buge
Epitech 3 – Promo 2004

Université catholique de Louvain

Laboratoire de télécommunications et télédétection

“ Benchmark de watermark ”

Sujet : “ Analyse, conception et mise en place d’un benchmark d’algorithme de watermark en ligne. Le but étant de simplifier les tests de la communauté scientifique, et le choix d’un algorithme adapté pour les utilisateurs désireux de protéger leurs créations numériques. ”

Du 01/05 au 30/08 2002

1. Sommaire

1. SOMMAIRE 1

2. RESUME 2

3. PRESENTATION DE L'ENTREPRISE 3

3.1. LE SECTEUR D'ACTIVITE 3
3.2. L'ENTREPRISE 4
3.3. LE SERVICE 6
3.4. LE POSITIONNEMENT DU STAGE DANS LES TRAVAUX DE L'ENTREPRISE 6

4. TRAVAIL EFFECTUE 7

4.1. LE CAHIER DES CHARGES 7
4.1.1. But général 7
4.1.2. Explication détaillée des résultats à obtenir 7
4.2. COMPTE-RENDU D'ACTIVITE 8
4.2.1. Axes d'étude et de recherche choisis 8
4.2.2. Déroulement concret des études 27
4.3. INTERPRETATION ET CRITIQUE DES RESULTATS 30

5. CONCLUSION GENERALE 31

6. BIBLIOGRAPHIE & GLOSSAIRE 32

7. ANNEXES 33

7.1. SOMMAIRE DES ANNEXES 33

2. Résumé

Le watermark est issue de la stéganographie, cette technique permet de dissimuler une information dans une autre.

On peut noter que l'application de la stéganographie est utilisable sur tout type de media : image, son, vidéo, modèle filaire…

Cette technique permet d'envoyer des informations via un canal insoupçonné par les personnes non averties.

L'utilisation du watermarking ne se limite pas à la communication de messages secrets. En effet la grande majorité des industries qui produisent des médias souhaitent utiliser cette technique afin de copywriter leurs œuvres de manière invisible de façon à prouver qu'elles sont bien les auteurs des documents watermarkés. Elles souhaitent également 'fingerprinter' les documents téléchargés par un utilisateur, de façon à faire respecter les droits de diffusion.

On se rend compte que la stéganographie peut être utile pour vérifier l'intégrité de documents mais ne résiste en aucun cas à la modification de ces derniers.

En effet dans le cas de la modification des valeurs d'un média, si nous appliquons une déformation ou un filtre au média, le message binaire sera détérioré.

Pour résister à ce genre d'attaque, il faut utiliser des algorithmes de watermarking robustes.

Ces algorithmes ne travaillent plus directement sur la représentation binaire du document, mais sur l'analyse de son contenu. Par exemple on utilisera pour l'image des algorithmes de modification de l'histogramme grâce aux transformées de Fourier, ce principe est également utilisable sur le spectre du son.

Les techniques modernes de watermark à base de clés permettent d'utiliser plusieurs fois le même algorithme dans un même document. Ces techniques servent également à empêcher les personnes ne disposant pas de clé, de lire le watermark.

Les algorithmes de watermark sont nombreux, les champs d'application le sont également :

En effet un producteur de film à besoin d'un watermark qui résiste à la compression MPEG, à la conversion digitale / analogique, à la conversion PAL / NTSC…

Un photographe à besoin d'un watermark résistant aux différents filtres connus, à la déformation, à la rotation, au découpage, au zoom, à l'impression, au scanne et aux divers algoritmes de compression destructeur…

De même pour un producteur de disque, qui a besoin d'un algorithme résistant au ré échantillonnage, au bruit, à la compression MP3…

Pour permettre aux scientifiques de tester leurs algorithmes, et aux industriels de choisir le watermark adapté à leur domaine, l'élaboration d'un benchmark est indispensable.

Dans ce rapport, nous détaillerons les démarches choisies pour le développement du benchmark, ainsi que les solutions techniques proposées pour mettre en œuvre ce projet avec les outils dont nous disposions.

3. Présentation de l’entreprise

3.1. Le secteur d’activité

La vocation du laboratoire de télécommunications de l'Université Catholique de Louvain est clairement expérimentale et ses opérations de recherche se développent autour de quatre thèmes :

Télédétection aux hyperfréquences

La télédétection est la discipline scientifique qui regroupe l'ensemble des connaissances et des techniques utilisées pour l'observation, l'analyse, l'interprétation et la gestion de l'environnement à partir de mesures et d'images obtenues à l'aide de plates-formes aéroportées, spatiales, terrestres ou maritimes. Comme son nom l'indique, elle suppose l'acquisition d'information à distance, sans contact direct avec l'objet détecté. Sa définition officielle est "l'ensemble des connaissances et techniques utilisées pour déterminer des caractéristiques physiques et biologiques d'objets par des mesures effectuées à distance, sans contact matériel avec ceux-ci".
Le laboratoire a réalisé des projets sur l'observation radar des forêts ou encore sur les perturbations pouvant être occasionnés par la pluie lors de l'observation de la surface de la mer.

Traitement d'images et codage

Le traitement d'images est la discipline du traitement de l'information qui vise à extraire et manipuler des informations de nature numérique et symbolique à partir d'images (éventuellement multiples) dans le cadre de systèmes autonomes ou d'aide à la décision. Les travaux du laboratoire dans ce domaine reposent sur la compréhension des images à partir de leurs formes et leurs structures, la compression d'images et de vidéos ainsi que le développement de systèmes de transmission d'images résistant aux erreurs.

Systèmes de communication numérique

Les activités de l'équipe sont centrées sur la conception et le développement de systèmes de télécommunication en totale conformité avec les normes actuelles. Il s'agit principalement d'optimisations globales de systèmes de communication visant à simplifier les systèmes, à améliorer la qualité d'une transmission ou à en corriger les défauts.

Réseaux multimédias sécurisés

Ce thème de recherche se propose de mettre au service des utilisateurs le fruit des études les plus récentes en matière de technologies de l'information de manière à réaliser des réseaux multimédias sécurisés comme des systèmes d'information médicaux, de transmission vidéo sécurisée ou de la réalité mixée et de la surveillance basée sur des systèmes multi agents.

3.2. L'entreprise

Historique

Fondée au quinzième siècle, le 9 décembre 1425, l'Université Catholique de Louvain est une des plus anciennes universités du monde. Son histoire est inséparable de celle de l'Europe, dont elle a partagé les mouvements de pensée et les troubles politiques. À la fin des années soixante, sous la pression de l'opinion flamande, la vieille université se sépare en deux sections.

En 1970, une loi accorde la personnalité civile à deux universités distinctes, l'une flamande qui reste à Leuven, l'autre francophone, qui émigre en Wallonie sur une terre de près de 1 000 hectares, une ville nouvelle qui prendra le nom de Louvain-la-Neuve. De 1972 à 1979, neuf facultés déménageront progressivement à Louvain-la-Neuve; la Faculté de médecine élira domicile à Bruxelles.

L'université Catholique de Louvain est une large communauté internationale : 20 000 étudiants de plus de 100 nationalités différentes, un staff de 5 000 enseignants, chercheurs et collaborateurs, 200 unités de recherche, 150 000 anciens dans le monde entier. C'est une université complète, qui forme près d'un universitaire sur deux en Belgique francophone, dans toutes les disciplines.

Le laboratoire de télécommunications

Le laboratoire de télécommunications et télédétection fait partie du département d'électricité de la faculté des sciences appliquées de l'Université Catholique de Louvain. Le responsable d'unité est Luc Vandendorpe, le président du département Piotr Sobieski et le doyen de la faculté est André De Herde.

Les activités de recherche du laboratoire structurent celui-ci en trois sous-groupes principaux :

- Le groupe Digicom dirigé par Luc Vandendorpe

Digicom travaille sur les systèmes de communications numériques

- Le groupe Télédétection dirigé par Piotr Sobieski

Son domaine de recherche est la télédétection aux hyperfréquences

- Le groupe Net-image dirigé par Benoît Macq

Il s'intéresse au traitement d'images, au codage et aux réseaux multimédias sécurisés

Le laboratoire fait également parti du Certi (Centre de recherche en technologies de l'information), fondé en 1996 dans le but de faciliter les contacts et les collaborations scientifiques dans ce domaine émergent. Les autres unités membres du Certi sont l'unité de dispositifs et circuits électroniques (DICE) et l'unité d'informatique (INFO).

Le personnel du laboratoire se répartit suivants trois statuts :

- le personnel académique (6 personnes)
- le personnel administratif, technique et ouvrier (8 personnes)
- le personnel scientifique (33 personnes)

Parmi le personnel scientifique, les assistants de recherche rémunérés par le budget de l'université partagent leur temps de travail en 50% de recherche et 50% d'encadrement didactique ; les autres sont rémunérés par des contrats extérieurs (projets européens, contrats région wallonne, contrats industriels privés). Le personnel scientifique est composé majoritairement de personnes suivant l'école doctorale, c'est un melting-pot, on rencontre des gens de toute nationalité (France, Italie, Espagne, Colombie, Japon ....)

Les projets européens

Les projets européens représentent une part importante de l'activité du laboratoire, ils sont financés par le programme relatif aux Technologies de la Société de l'Information appelé IST (Information Society Technologies) qui dispose d'un budget de 3.600 millions d'Euro, ce programme est géré par la commission européenne.

Ce programme est consacré à l'usage des technologies dans la société de l'information, il fait partie du 5ème programme-cadre de recherche, de développement technologique et de démonstration (RDT) européen. La Société de l'Information est la réponse de l'Union européenne à la révolution sociale engendrée par les développements récents et accélérés dans le secteur des technologies de l'information et de la communication. Ces projets permettent de promouvoir les travaux du laboratoire, de faciliter les échanges scientifiques et de financer les assistants de recherche.

Les Spin-off

Une Spin-off est une société créée par des chercheurs de laboratoire qui exploitent les résultats de leurs recherches et des technologies qu'ils ont mis en œuvre. Le laboratoire de télécommunications de l'UCL est un incubateur de spin-off, en multipliant les partenariats avec des entreprises et en participant à de nombreux projets européens, il a gagné la confiance des industriels et des institutions académiques ce qui lui a permis de récolter des fonds pour créer des sociétés. Plusieurs sociétés ont été fondées par des chercheurs du laboratoire, on compte parmi celles-ci Telemis, Alterface et Octalis.

Telemis est une société qui développe des logiciels dédiés à la médecine et la radiologie. Elle met en place une technologie réseau et informatique permettant l'échange interactif d'informations médicales d'imagerie entre plusieurs institutions.

Alterface est une spin-off située dans les locaux du laboratoire de télécommunications, elle travaille sur la réalité virtuelle mélangée qui permet de faire intervenir par exemple des personnes dans un monde virtuel par le biais de caméras.

Octalis fournit des solutions d'accès conditionnel et de transmission vidéo sécurisée, cette spin-off a été lancée par 5 chercheurs.

3.3. Le service

Durant mon stage, j'ai intégré le groupe Net-image du laboratoire de télécommunication et télédétection de l'université catholique de Louvain dirigé par monsieur Benoît Macq. J'ai été intégré dans une nouvelle équipe, dont la tâche était de mettre au point un benchmark de watermarking. Cette équipe est formée de trois personnes :
Mon maître de stage : Frédéric Lefebvre, responsable du projet.
Audric Thevenet, stagiaire d'Epita, s'occupant du développement de l'interface graphique, du déploiement sécurisé du benchmark ainsi que les composants logiciels dont dépend ce dernier. Et moi-même. Ma tâche consistait à l'analyse et au développement du noyau du benchmark.

3.4. Le positionnement du stage dans les travaux de l'entreprise

L'élaboration du benchmark permettra de délester la charge de développement au créateur d'algorithmes de watermark : en effet le code de chargement et de modification des médias ne sera pas à écrire, il sera centralisé pour tous les utilisateurs. La génération des résultats pour le grand publique sera également automatisée. Ceci supprimera les tâches communes et répétitives relatives à l'élaboration et au test de tous les algorithmes de watermarking.
Ce qui permettra aux équipes de chercheurs de l'U.C.L. de se concentrer exclusivement sur le développement de l'algorithme proprement dit. Cet outil ne pourra donc qu'améliorer les temps de développement et la qualité des algorithmes.

4. Travail effectué

4.1. Le cahier des charges

4.1.1. But général

Le travail demandé consiste à élaborer un benchmark pour les algorithmes de watermarking d'image fixe.

4.1.2. Explication détaillée des résultats à obtenir

Ce benchmark devra être utilisable via une page internet et permettra de comparer les algorithmes entre eux. Il devra être simple d'utilisation (élaboration d'une documentation utilisateur)

Le benchmark devra permettre au développeur de tester ses algorithmes avant de rendre les résultats publics. Il devra également permettre de tester des algorithmes autonomes écrits pour les plates-formes de développement les plus répandues. Le benchmark devra reprendre les principales fonctionnalités des benchmarks existants

4.2. Compte-rendu d'activité

4.2.1. Axes d'étude et de recherche choisis

La palette des utilisateurs du benchmark doit être la plus large possible : en effet elle doit regrouper la communauté scientifique et les entreprises développant les algorithmes. Elle doit aussi inclure tous les types d'utilisateur de ces technologies : entreprises, ou utilisateurs privés.

L'hétérogénéité des environnements des utilisateurs nous a poussé à choisir une interface WEB pour le benchmark. Ce choix permet de délester l'utilisateur de l'installation du benchmark sur une machine locale, et nous permet de développer une interface fonctionnant sur toutes les plates-formes répandues dans le grand public ou dans les petites et grandes entreprises. Ceci nous permet également de centraliser les composants logiciels et matériels du benchmark, ce qui facilite les mises à jours éventuels.

Toutefois, il faudra prévoir la mise en place d'un système permettant aux utilisateurs de compléter eux-mêmes le benchmark.

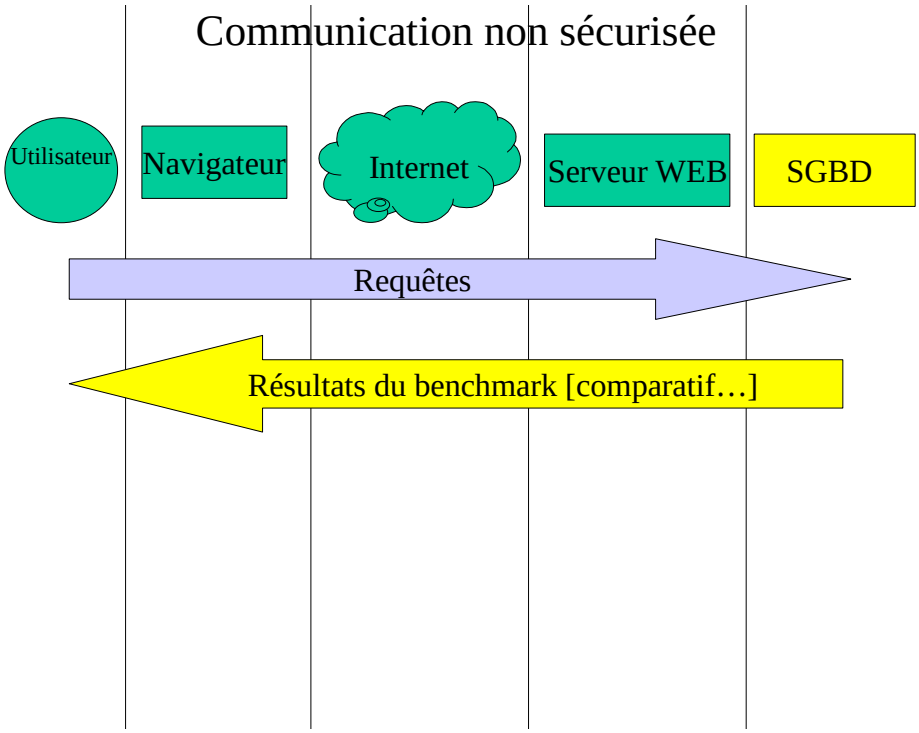
Le Benchmark doit être complètement libre de droit. En effet son élaboration est motivée par les chercheurs de l'U.C.L. qui souhaitent mettre cet outil à disposition du plus grand nombre. C'est pourquoi l'utilisation de logiciels libres et de l'Open Source à été l'orientation décisive pour le choix des technologies.

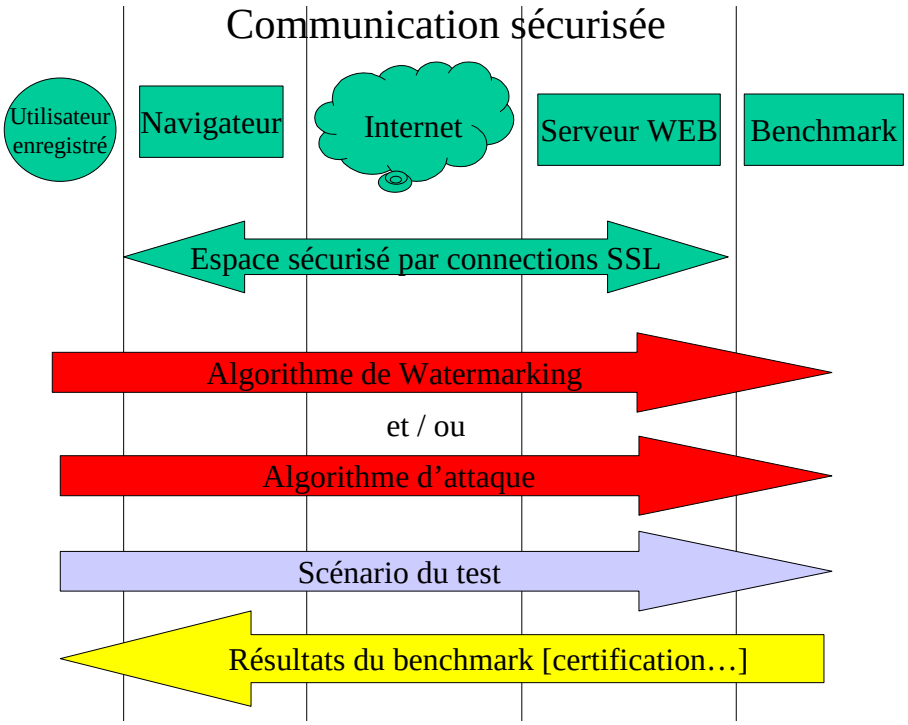
En effet l'interface du benchmark à été installée sur une machine de type PC sous un système d'exploitation NetBSD. Le serveur WEB utilisé est Apache. Les scripts CGI nécessaires pour l'interaction entre l'interface graphique et le noyau du benchmark ont été écrits en PHP.

Pour le stockage des requêtes des utilisateurs et des résultats du benchmark, l'utilisation d'un S.G.B.D. nous parut la solution la plus intéressante en terme de temps de développement. Pour rester dans le domaine du logiciel libre, le choix de MySQL fut indiscutable.

La sécurité de la communication est également nécessaire, car avec la solution choisie, les personnes désirant tester leurs algorithmes sont obligées de les " Uploader ".

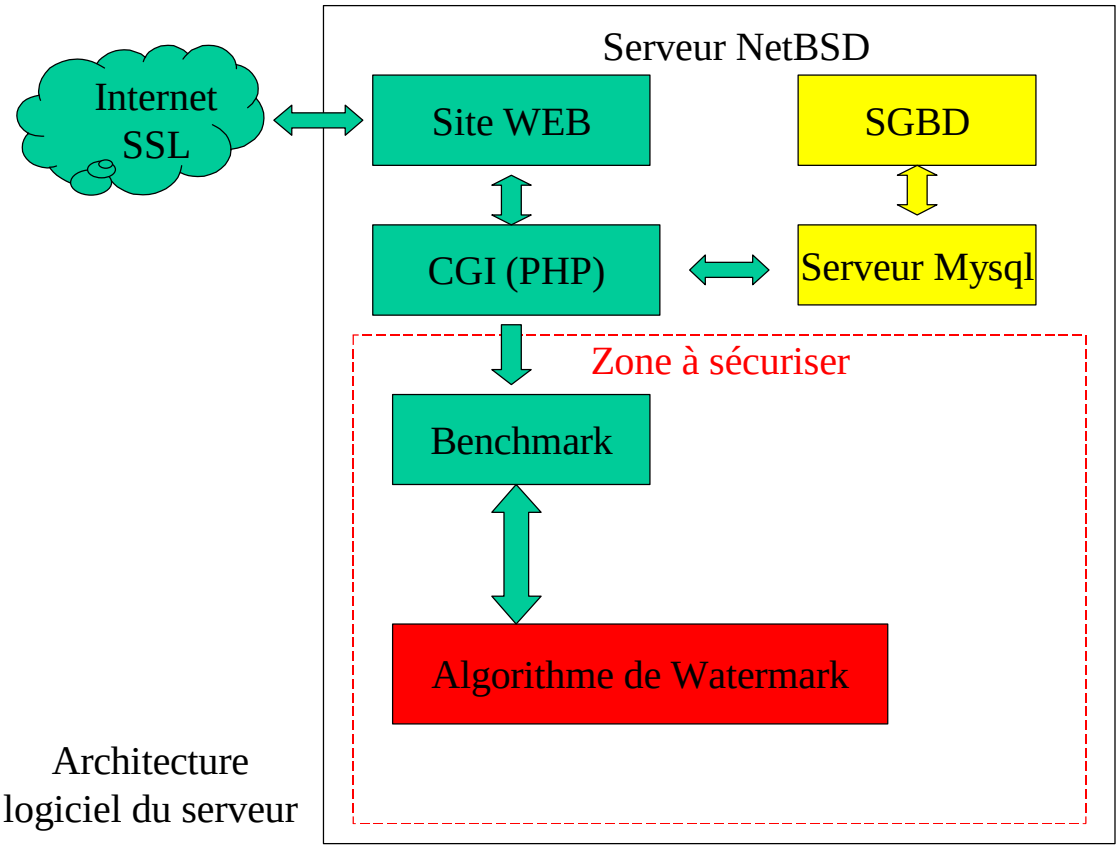
Ceci pause un problème car les algorithmes sont protégés par des brevets : le secret industriel intervient lors de l'élaboration de ces technologies. La diffusion à grande échelle d'un algorithme " Uploadé " serait une catastrophe pour l'entreprise désirant tester ses recherches avec notre outil. C'est pourquoi nous avons utilisé Open SSL pour crypter les communications entre le site et ses utilisateurs.


Communication non sécurisée
Utilisateur
Navigateur
Internet
Serveur WEB
SGBD
Requêtes
Résultats du benchmark [comparatif…]


Communication sécurisée
Utilisateur enregistré
Navigateur
Internet
Serveur WEB
Benchmark
Espace sécurisé par connections SSL
Algorithme de Watermarking
et / ou
Algorithme d'attaque
Scénario du test
Résultats du benchmark [certification…]

Le noyau du benchmark a été écrit en C car ce langage est très répandu dans le monde informatique et qu'il est très utilisé au sein de l'UCL. Toutefois, malgré l'âge de ce langage, un effort de développement moderne orienté objet a été observé de manière à rendre le code lisible et modulaire. Car rappelons le, ce code est Open Source et donc destiné à être lu par un large public.

Le rôle du noyau du benchmark est d'interpréter les scénarios de tests, d'exécuter les algorithmes de watermarking, de les attaquer par différentes méthodes et de stocker les résultats des tests dans la base de donnée.



Architecture logiciel du serveur

Axes de recherche pour le scénario

Le scénario de test doit permettre à l'utilisateur de personnaliser les différentes étapes nécessaires à l'application d'un algorithme de watermarking, tout en pouvant placer des points de contrôle de robustesse et des mesures de qualité à tout moment. Il doit aussi pouvoir configurer la liste et le paramétrage des attaques à exécuter ainsi que les médias choisis.

Voici donc les 5 commandes qu'un scénario doit être capable d'interpréter :

- le chargement des médias nécessaire pour le test.
- l'application de l'algorithme de watermark sur un média choisi.
- le lancement éventuel de une ou plusieurs attaques sur le ou les médias watermarkés.
- le lancement, en certains points critiques du scénario, de l'algorithme de relecture de la marque sur un média ; ceci pour vérifier la robustesse du watermark, ou tout simplement, son bon fonctionnement.
- Le lancement éventuel de un ou plusieurs algorithmes mesurant les critères de qualité.

Pour des raisons de simplicité d'utilisation et de portabilité, le scénario sera rédigé en A.S.C.I.I.. Il pourra être saisi aussi bien sous un éditeur Unix que Windows.

Un système de variable devra être mis en place pour permettre l'identification des médias. En effet, un scénario devra permettre de watermarquer plusieurs médias et d'effectuer plusieurs traitements d'affilés sur le même média. Ce système permettra les combinaisons d'attaques.

Dans un premier temps l'interprétation du scénario se fera séquentiellement. Cette méthode réduira le temps de développement. L'utilisation du système de variable permettra une future évolution vers des scénarios avec structures de boucles et branchements conditionnels.

Les cinq commandes auront bien entendu des arguments obligatoires nécessaires à leurs exécutions. Mais un nombre variable d'arguments optionnels pourra être inséré à la suite, permettant ainsi une plus grande flexibilité au niveau du paramétrage des algorithmes appelés.

Pour les arguments optionnels, une convention d'appel par nom sera adopté afin d'autoriser leur omission, ceci permettra également de faire abstraction de leur ordre lors de l'appel.

Développement technique de l'interpréteur de scénario

Le scénario est une suite de commande. Une commande est contenue sur une seule ligne, elle est composée de plusieurs mots séparés par une suite d'espaces ou de tabulations dont voici la syntaxe :

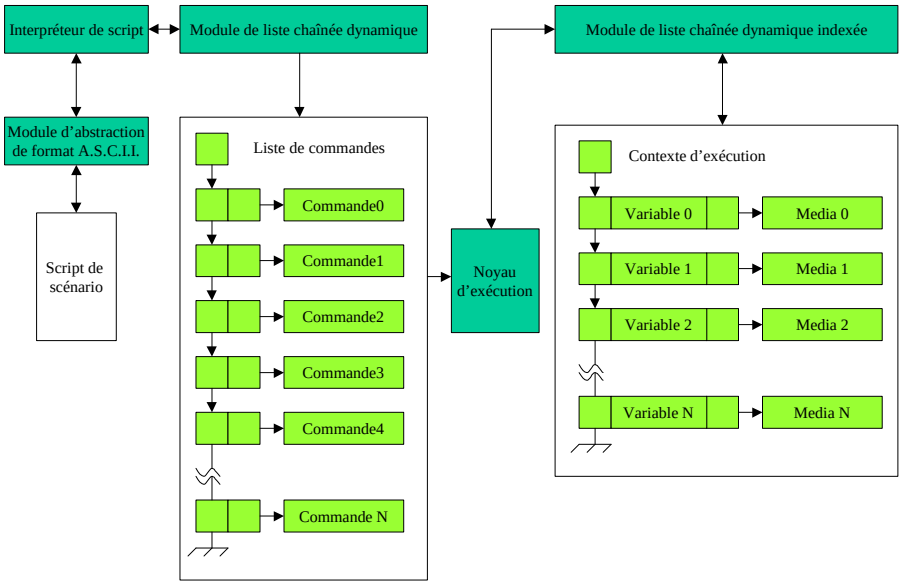
Nom_commande [Argument_obligatoire[…]] [Nom_argument Argument_facultatif […]]

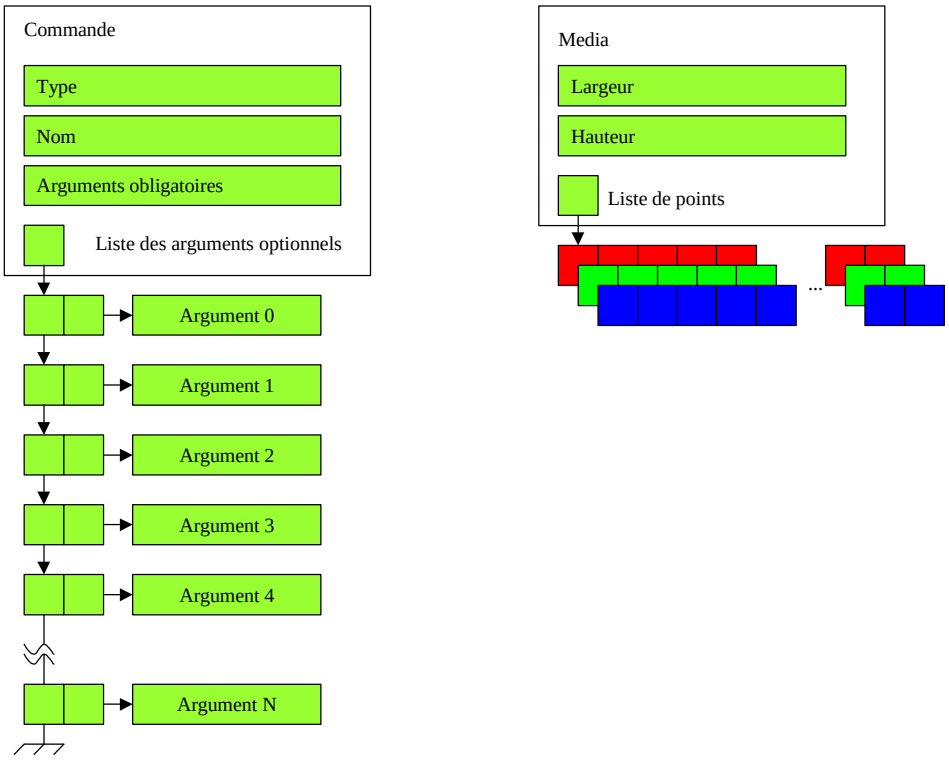
La première étape lors de l'interprétation d'un fichier de scénario est de déterminer si la syntaxe de la ligne de commande est correcte, pour cela, l'interpréteur devra vérifier si le nom de la commande fait bien partie des 5 connues. Dans un deuxième temps il devra vérifier si le nombre d'argument passé est supérieur ou égal au nombre d'argument obligatoire attendu. Ce nombre dépend de la commande spécifiée. Il devra enfin vérifier si le nombre d'argument optionnel est bien paire (c'est a dire qu'il y a bien une suite de couple Nom-Valeur pour chaque argument optionnel)

La deuxième étape de l'interprétation est de stocker les commandes dans une structure en mémoire facilitant leur exploitation.

Un module de liste chaînée dynamique de pointeur sera créé à cet effet, ce qui permettra tout d'abord de stocker en mémoire une suite de structures représentant les commandes à exécuter.

Un autre module sera créé : celui de liste chaînée dynamique de pointeur indexé. Il " dérive " directement du précédant et permettra de stocker la liste des arguments optionnels pour chaque commande. L'index dans notre cas correspondra au nom d'un argument ce qui assurera l'unicité de l'argument lors de l'affectation de sa valeur. Ce système permettra également de savoir si oui ou non un argument optionnel a été affecté lors de la commande.





Certain arguments obligatoires seront des noms de variable interne a l'interpréteur. A ces noms de variable correspondra la représentation standard d'un média en mémoire.
Ce qui permettra à la personne rédigeant le scénario de spécifier quelles données passer à l'algorithme.

L'interpréteur devra donc contrôler qu'une variable attendue en entrée d'une commande ait préalablement été affectée à un média, que les anciennes ressources allouées à la variable attendue en sortie d'une commande soient correctement libérées avant d'être écrasées, ou tout simplement créer les variables de sortie si elles n'existent pas déjà dans le contexte d'exécution. Le module de liste chaînée dynamique de pointeur indexé utilisé pour les arguments des commandes répondra parfaitement à cette attente. Il suffira simplement d'utiliser une seule liste de ce type pour créer un contexte durant toute l'exécution du scénario. Le nom des variables servira d'indexe.

Axes de recherche pour les fichiers de configuration

Le fichier de configuration doit permettre de configurer totalement le benchmark de manière à l'installer sur des systèmes totalement hétérogènes. Ce fichier permettra à l'administrateur du site de configurer l'emplacement des médias, de spécifier la machine contenant le serveur MySql et définir les paramètres de connexion (loggin, password, nom de la base de donnée…) il pourra également spécifier le numéro du port d'écoute du serveur de benchmark. Ce port servant à recevoir les ordres de l'interface WEB.

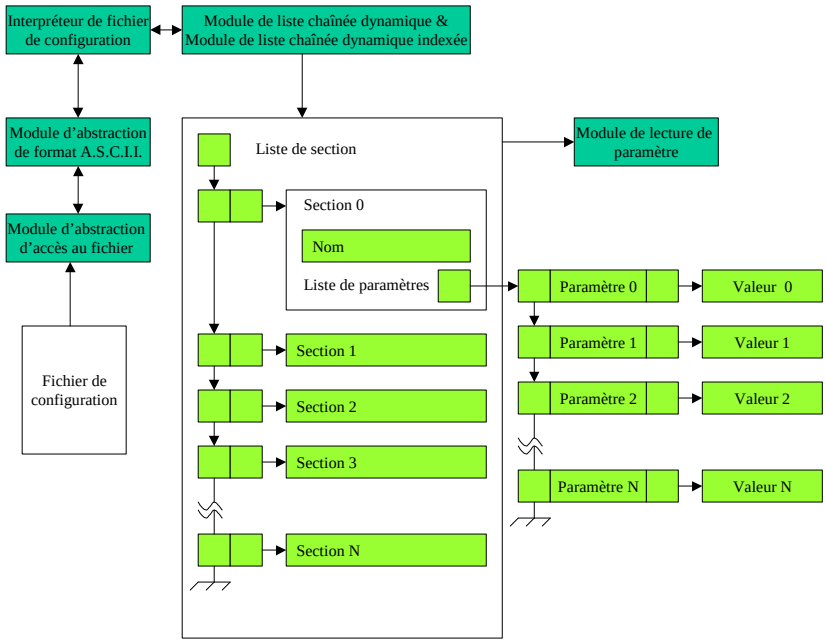
Dans cette organisation répartie du benchmark, un fichier de configuration est nécessaire. Cette flexibilité permettra à l'administrateur de déployer les différentes parties du benchmark sur les machines qu'il souhaite : Interface WEB, Serveur MySQL, Serveur de benchmark.

Développement technique des fichiers de configuration

L'architecture des fichiers de configuration est très proche de celle des scénarios, ce qui permettra de réutiliser les modules préalablement développés :

Le fichier de configuration est un fichier A.S.C.I.I. saisit sous Windows ou Unix, il est découpé en section. Chaque section commence par une chaîne du type [nom_de_section] précédée d'une suite de ligne contenant chacune, une affectation du type nom_param = valeur.

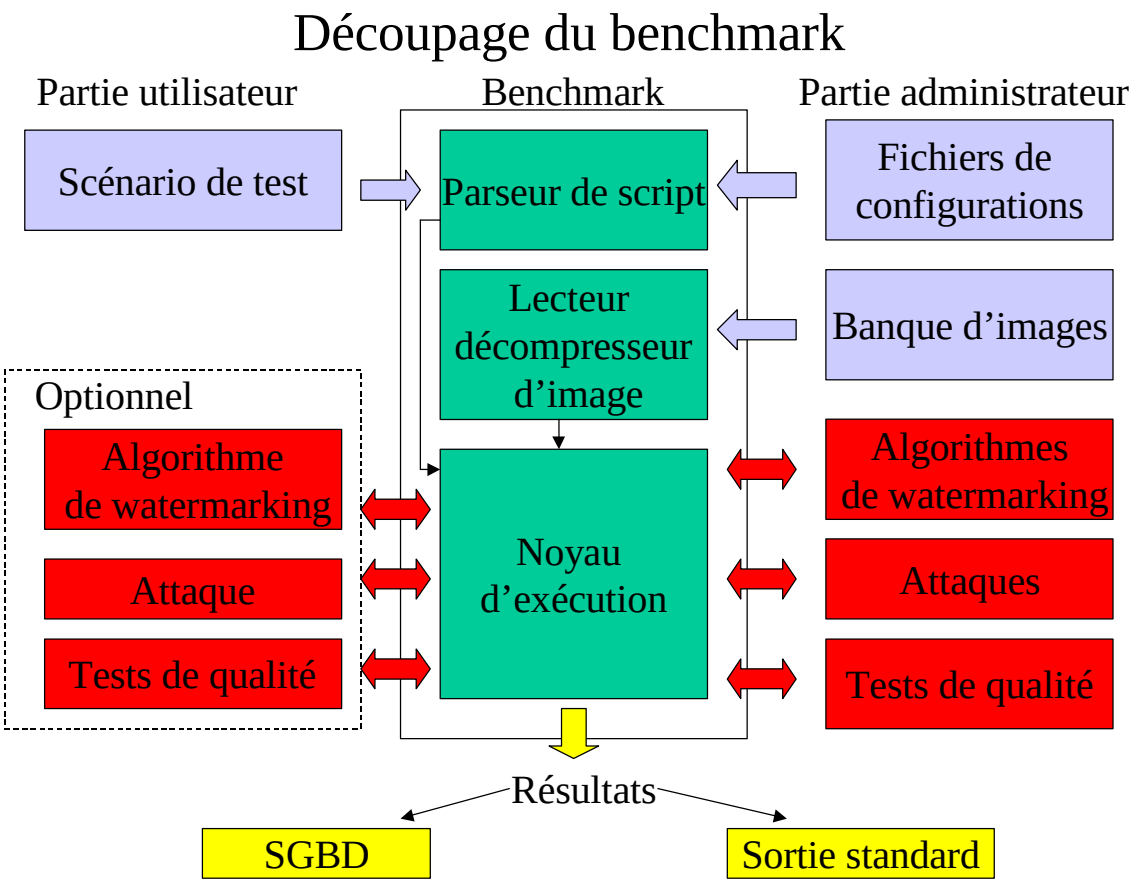
La structure mémoire d'un fichier de configuration sera donc une liste chaînée dynamique de sections contenant chacune, une liste chaînée dynamique indexée de paramètres. Ceci permettra l'unicité des affectations de paramètre dans chaque section, mais également de pouvoir créer des paramètres de même nom dans des sections différentes. Ceci sera très utile pour les futures évolutions.



Axes de recherche pour les algorithmes

Pour permettre l'évolutivité et pour réduire le temps de développement des utilisateurs, le système de watermarking à été découpé en quatre grands groupes de librairies utilisateurs toutes indépendantes et interchangeables :

- Les librairies de chargement de média
- Les librairies de watermarking
- Les librairies d'attaque
- Les librairies de test de qualité.

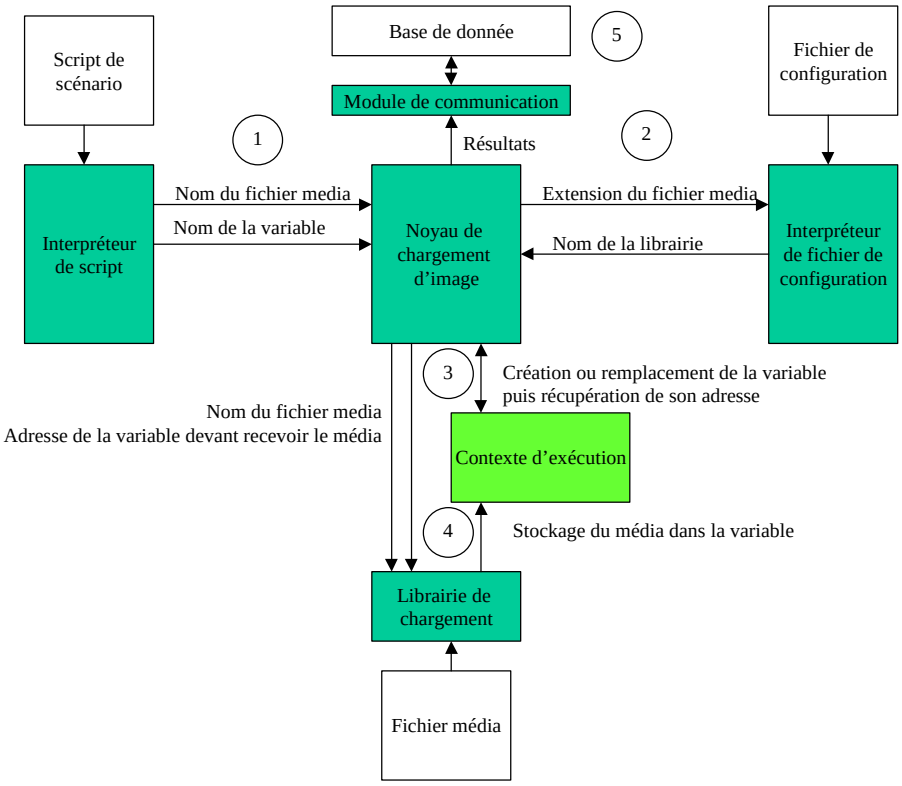


Librairie de cbargement de média

Le rôle des librairies de chargement de média est de transformer un format de fichier contenant un média dans une structure en mémoire exploitable par le benchmark, ce type de librairie est le seul développé en interne et que l'utilisateur n'a donc pas a uploader :

En effet il a été décidé que le site de benchmark de watermark ne devait en aucun cas servir de site de watermarking. C'est pourquoi l'utilisateur ne peut pas uploader les médias equ'il veut watermarker, mais seulement les choisir dans une liste fournie par le serveur de benchmark. Cette méthode limite l'utilisation abusive des algorithmes de watermarking présent sur le site. Les formats de fichier contenant les médias sont donc connus et choisis par les administrateurs du site. Ne connaissant pas au préalable le format de fichier utilisé au final par l'UCL, il nous parut préférable d'utiliser un système de librairie permettant, par simple modification du fichier de configuration, d'affecter une extension de fichier à une librairie de chargement de média. Cette méthode est d'autant plus flexible car elle permet de rajouter, voir modifier, des formats de média tout au long de la vie du site, et cela de façon totalement transparente pour l'utilisateur.

Chargement d'un fichier média

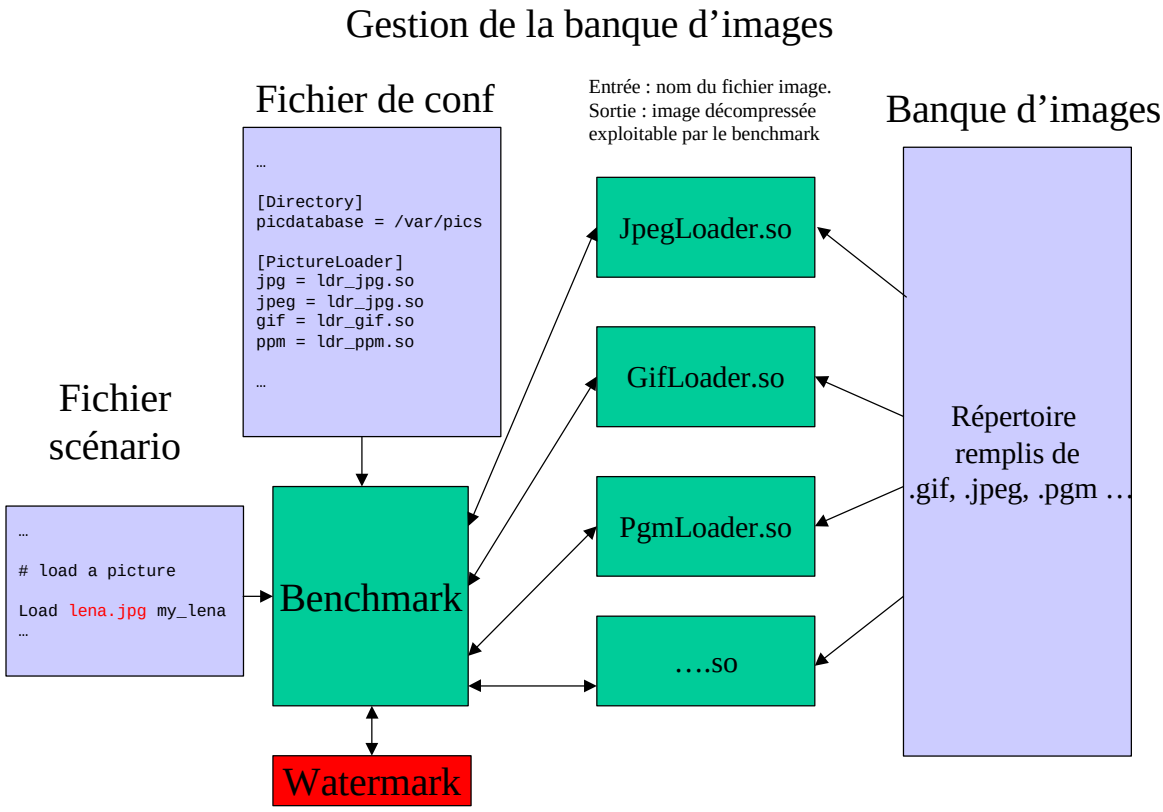


Développement technique de l'interprétation d'une commande de chargement de media

Lorsque l'interpréteur rencontre la commande load, il sait donc qu'il doit trouver deux arguments obligatoires : le premier est le nom du fichier contenant le média, le deuxième est la variable recevant la structure standard correspondant au média.

Lors de l'exécution de cette commande, l'interpréteur devra dans un premier temps lire dans le fichier de configuration l'emplacement des fichiers médias. Puis une fois le chemin complet créé par simple concaténation, il devra identifier l'extension du fichier media puis chercher dans le fichier de configuration le nom de la librairie de chargement de média susceptible de lire le contenu du fichier media. Si cette librairie n'existe pas ou si son exécution échoue (fichier média introuvable ou corrompu), l'utilisateur devra en être informé, le temps de lecture du média sera conservé a titre d'information.

La variable associée à la structure standard du média est a présent chargée lors de l'exécution de la librairie si cette dernière à des points d'entrée corrects, si ce n'est pas le cas, l'utilisateur sera également informé si la librairie n'existe pas ou est corrompue.



Axe de recherche pour une librairie de watermarking

Une librairie de watermarking doit contenir au moins deux fonctions :

- une fonction qui appliquera l'algorithme de watermarking sur le média spécifié en fonction d'une liste d'arguments.
- une fonction qui retournera les données watermarkées dans un document spécifié toujours en fonction d'une liste d'arguments.

Développement technique

L'appel de cette librairie se fera explicitement dans le fichier de scénario. La syntaxe sera la suivante :

watermark nom_algo media_in media_out data [argument valeur…]

la commande watermark prendra trois arguments obligatoires : le nom de l'algorithme de watermarking (correspond directement au nom de la librairie), la variable correspondant au media a modifier, la variable qui recevra le média watermarké et la donnée à insérer dans le média.

Les autres arguments de la commande seront passés à l'algorithme de watermarking sans aucune interprétation de la part du benchmark.

Un second type d'appel de la librairie est possible :

Wm_decode nom_algo media_in data [argument valeur…]

Cette commande exécutera l'algorithme de décodage sur le média spécifié, et vérifiera que la marque lue est bien égal à la donnée passée en argument.
Les librairies d'attaque se comporteront comme les librairies de watermark : elles liront en entrée un média, elles retourneront en sortie un média modifié. Elles prendront aussi un nombre variable d'argument. Leur appel sera également explicite dans le fichier de scénario grâce a la commande attack.

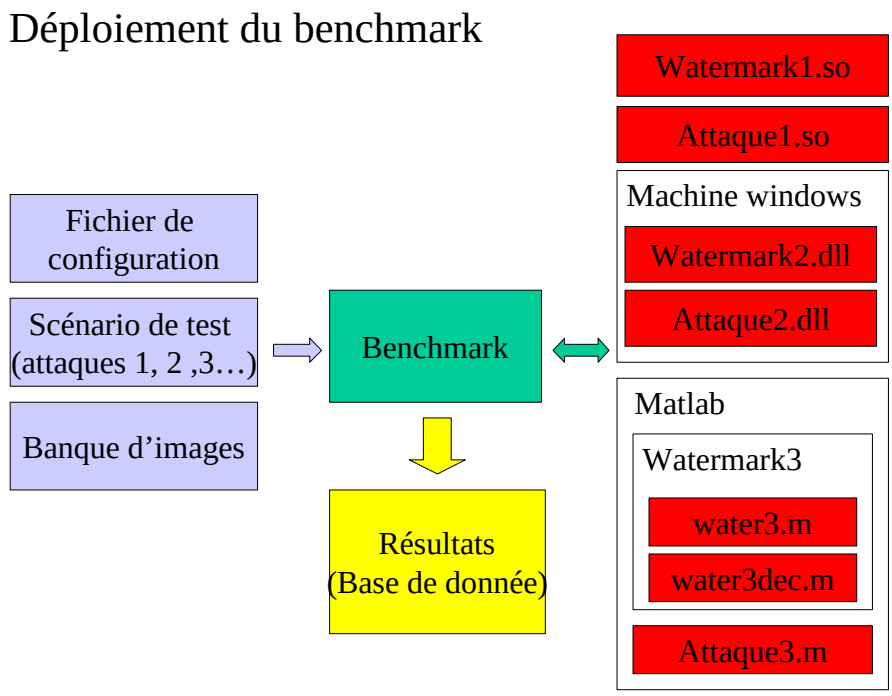
Les librairies de test de qualité prendront deux médias en entrée, et retourneront des coefficients de qualité entre ses deux médias. L'appel sera effectué par la commande test. Dans tous les cas, le temps d'exécution des librairies sera conservé a titre indicatif.

Librairies et plates formes hétérogènes

L'utilisation de librairie peut poser des problèmes car elles sont compilées dans le langage machine spécifique au processeur utilisé, elles sont également structurées en fonction du système d'exploitation utilisé. Ceci pause un gros problème de portabilité qui risque de fortement gêner l'utilisateur du benchmark. En effet, nous ne pouvons pas nous permettre de lui imposer un système d'exploitation pour tester ses algorithmes, et encore moins l'architecture matériel.

C'est pourquoi il nous faut découper le serveur de benchmark en plusieurs parties :
En effet, il faut mettre d'un coté la partie serveur de benchmark, qui interagie avec l'interface graphique, qui interprète les fichiers de scénario et qui charge les médias.
L'autre partie sera un serveur de librairie, qui se contentera d'exécuter les librairies de son architecture en fonction des ordres du serveur de benchmark et qui retournera les résultats.

En fait il faudra développer un serveur de librairie par plate-forme utilisée.
L'élaboration d'un protocole permettra l'abstraction de la plate-forme et du type de librairie utilisée. Tout le monde pourra donc écrire des serveurs de librairie complètement différents à condition de respecter le protocole réseau établi.



Ce système permettra également d'installer des serveurs de librairie différents sur la même plate-forme. Ce qui sera nécessaire dans notre cas.

Cette méthode permettra de faire varier le format des librairies, ce qui permettra dans le futur de développer des serveurs de librairie capables d'exécuter des librairies créés pour d'autres benchmarks que celui de l'U.C.L. par exemple les librairies de Stirmark.

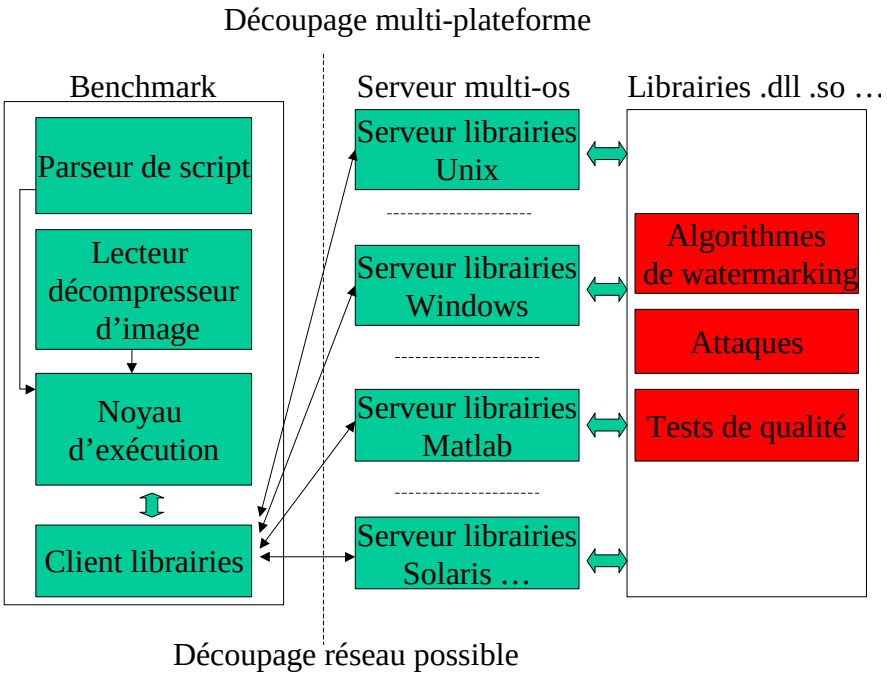
Ceci va également permettre d'utiliser des formats de librairie écrits en d'autre langage comme par exemple, en Matlab. Ce langage de programmation scientifique est très utilisé dans la recherche, il est donc notre devoir de permettre à tous les utilisateurs de ce langage interprété d'utiliser notre benchmark.

Toutefois nous ne perdons pas de vue la création de librairie compilée qui est également très répandue et beaucoup moins onéreuse en terme de licence. C'est pourquoi il nous faudra développer des serveurs de librairie pour les plates-formes et les systèmes les plus répandus.

Développer chaque serveur de librairie compilé séparément aurai été une tâche très longue et rébarbative, c'est pourquoi nous avons préféré garder un seul " source tree " pour tout le code du benchmark, quelque soit la plate-forme utilisée.

Toutefois, il a fallut revoir le code source existant et prévoir le futur en macro définissant les appels systèmes utilisés et en les encapsulant dans des nouveaux modules pour simplifier le code appelant.

Par exemple l'interpréteur générique de fichier A.S.C.I.I. utilisait les commandes de mappage de fichier, il fallut écrire un module multi plate-forme de mappage de fichier pour éviter que l'interpréteur ait à utiliser les appels systèmes.
Il fallut également créer un module multi plate-forme de socket réseau, un module multi plate-forme de chargement de librairie dynamique ainsi qu'un module multi plate-forme de chronométrage.



Unix est un système orienté multi processus et Windows et un système orienté multitâche, ceci complique l'écriture des serveurs réseau. C'est pourquoi il fallut également écrire un module d'abstraction supplémentaire pour les traitements parallélisés évitant ainsi l'utilisation directe de l'appel système fork dans les programmes. En Effet, sous les plates-formes Windows, fork n'existe pas et devra être remplacé par un mécanisme multithread nécessaire au traitement simultané des connections clientes.

Deux autres problèmes ont également du être résolus, notamment le système d'adressage sur 32 ou 64 bit lors de l'utilisation des pointeurs, mais également l'ordonnancement des bits dans les registres du processeur (Little Indian/ Big Indian). Un module de communication bâti sur le module de socket réseau fait abstraction de cette particularité architecturale.

Le serveur de librairie compilé pourra ainsi être compilé sur les plates-formes suivantes :

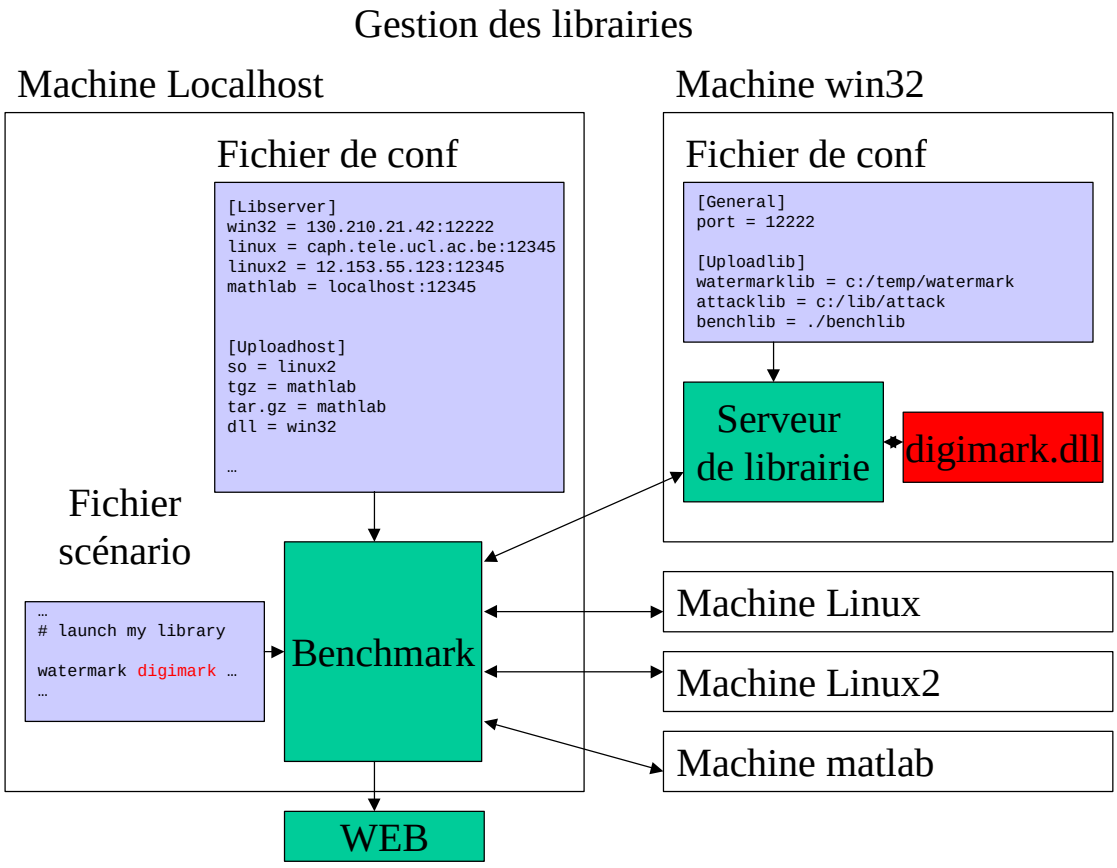
- Linux debian /PC
- NetBSD / PC
- Windows 2000 / PC
- OSF / Alpha
- Solaris / Sun

Il est possible que le serveur compile sur d'autres plates-formes ou leurs dérivées, mais celles citées ci-dessus ont été les seules à être testées

Mettre en place une telle infrastructure oblige le serveur de benchmark a connaître tout les serveurs de librairie disponibles sur le réseau. Pour cela, nous avons complété le fichier de configuration de manière à se qu'il répertorie le nom des machines et les ports d'écoute de tous les serveurs de librairie installés. Cette section permettra encore plus de flexibilité au niveau de la configuration et de l'évolution du benchmark.

Un autre problème posé par ce découpage réseau du serveur de benchmark est la complication de l'upload d'algorithme. En effet le serveur de benchmark reçoit des librairies qu'il n'est plus capable d'exécuter, il faut qu'il upload a nouveau la librairie sur la machine hébergeant le serveur de librairie susceptible de remplir cette tâche.

La solution proposée à ce problème à été de regarder à chaque upload, l'extension du fichier librairie, et de lire dans le fichier de configuration, la machine ou se trouvait le serveur de librairie correspondant.

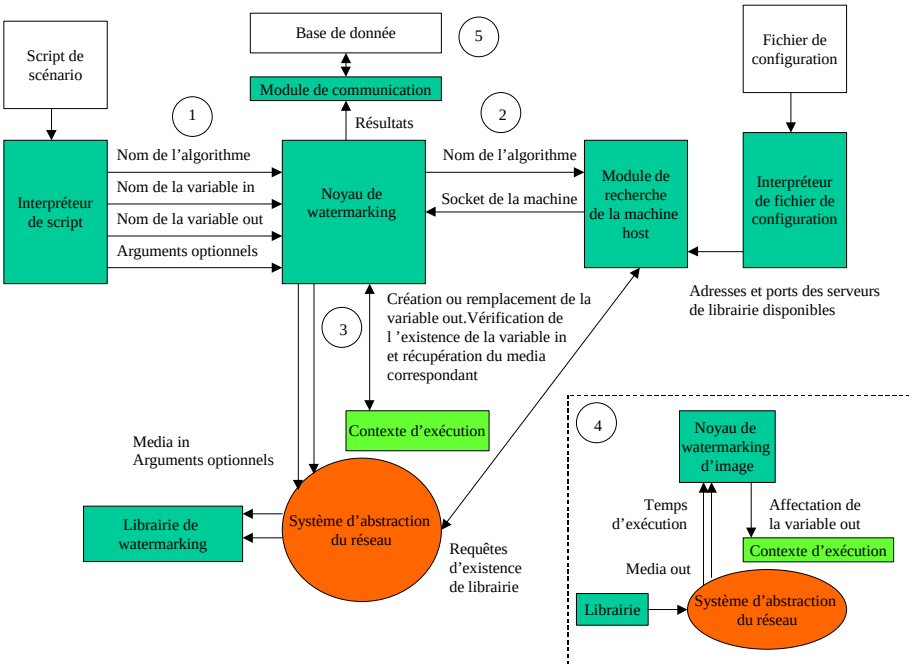


Pour l'exécution et la suppression des librairies, le même problème se pause. Les librairies n'étant plus présentent sur la machine du serveur de benchmark, il faut qu'il puisse demander au bon serveur d'exécuter la librairie. Une fonction du protocole réseau permettra de déterminer l'existence d'une librairie sur le serveur de librairie spécifié. Le serveur de benchmark n'aura plus qu'à interroger tous les serveurs de librairie connus jusqu'à ce qu'un d'eux réponde présent.

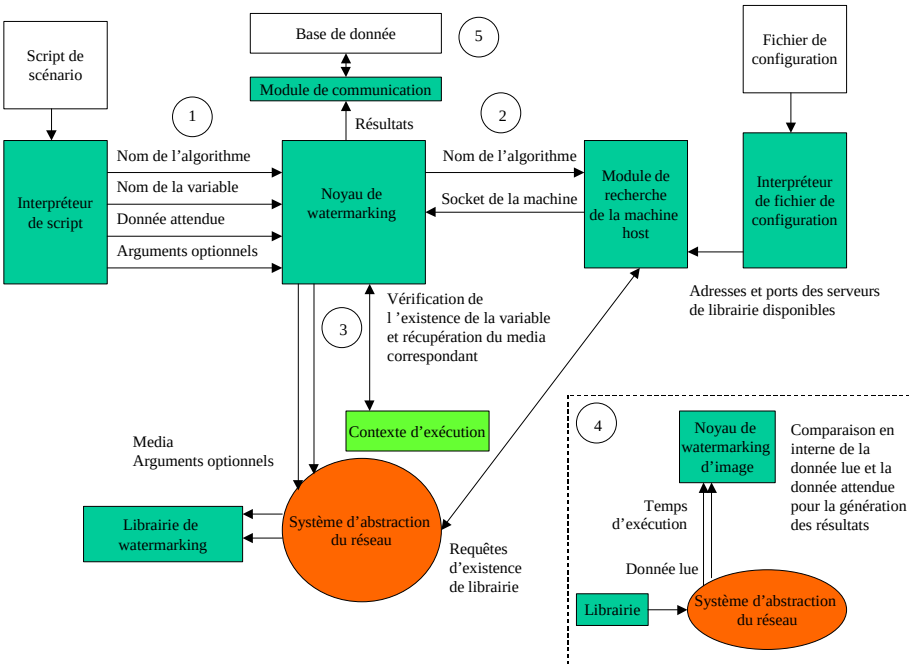
La tache des serveurs de librairie est donc multiple : ils doivent permettre d'exécuter 3 types de librairie : watermark, attaque, test, mais il doivent aussi lire et répondre aux ordres du réseau, télécharger des nouvelles librairies ou les supprimer.

C'est pourquoi il nous parut plus souple d'y inclure le même système de fichier de configuration que pour le serveur de benchmark, ce qui permettra de configurer le port d'écoute et les répertoires de réception des 3 types de librairie et d'éventuelles spécificités liées aux plates-formes d'accueil.

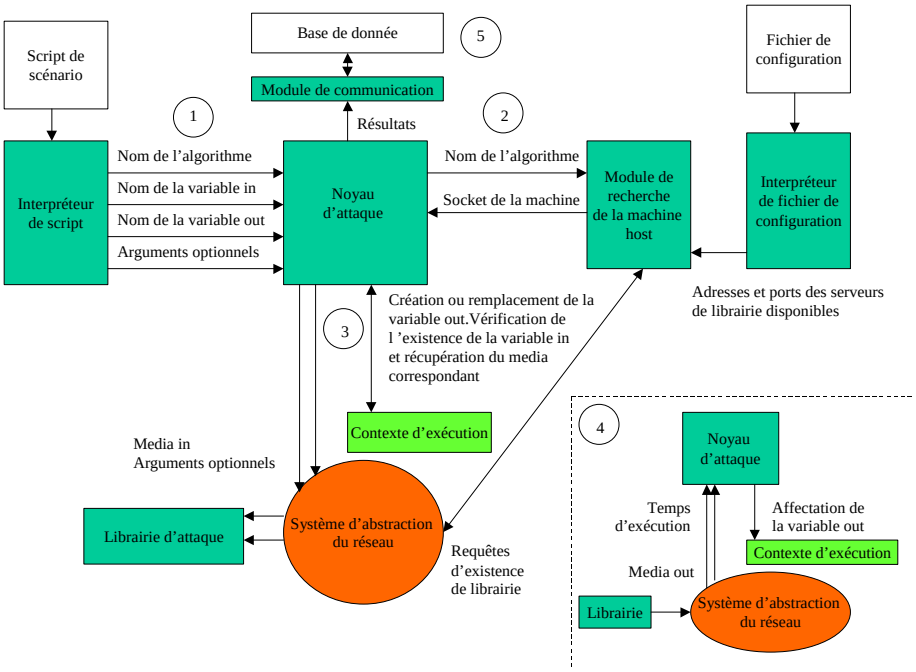
Watermarking d'un média



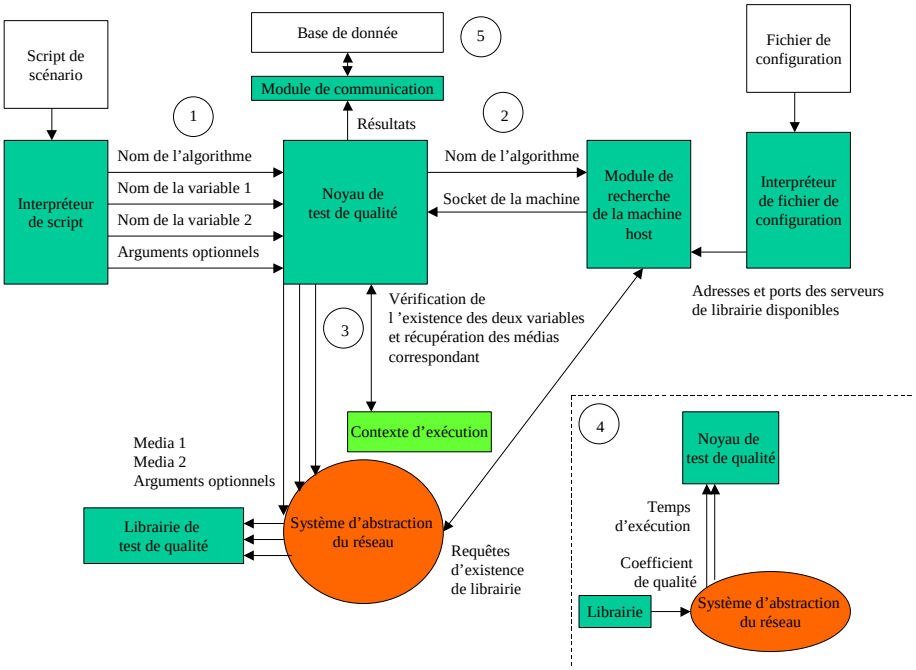
Lecture de la marque contenue dans un média



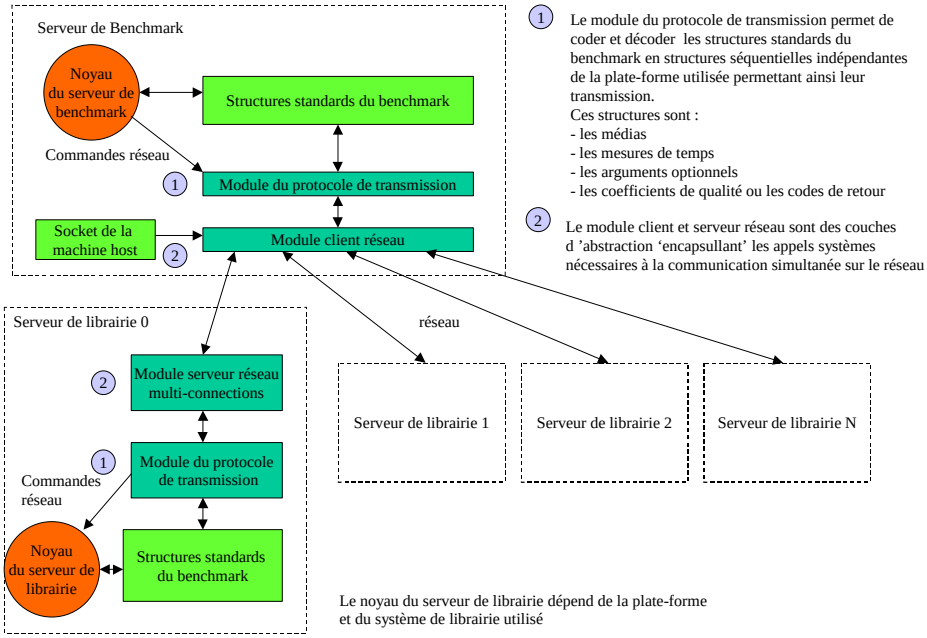
Attaque d'un média



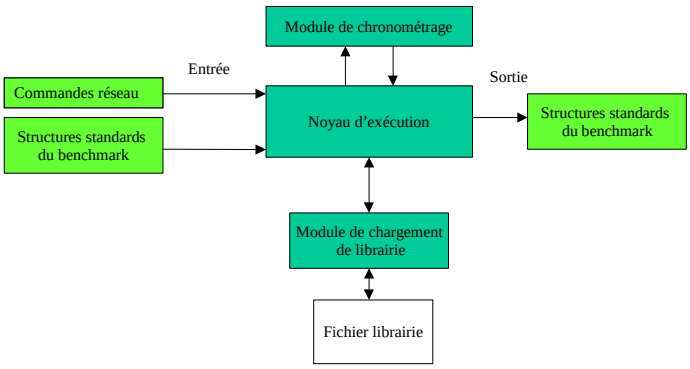
Test de qualité entre deux médias



Système d'abstraction réseau



Noyau du serveur de librairie native (Win32, Linux, Unix...)



Serveur de librairie Matlab

Le serveur de librairie Matlab est un peu plus complexe que les serveurs de librairie compilée. En effet sous Matlab, chaque fonction est écrite dans un fichier A.S.C.I.I. distinct portant son nom. Ce qui implique qu'une librairie peut être formée de plusieurs fichiers (ce qui est forcement le cas des librairies de watermarking).

Ceci pause un problème de conception pour l'upload de librairie. C'est pourquoi il a été décidé d'archiver les fichiers sources des librairies Matlab dans une archive. C'est ce fichier archive qui représentera la librairie pour le serveur de benchmark.

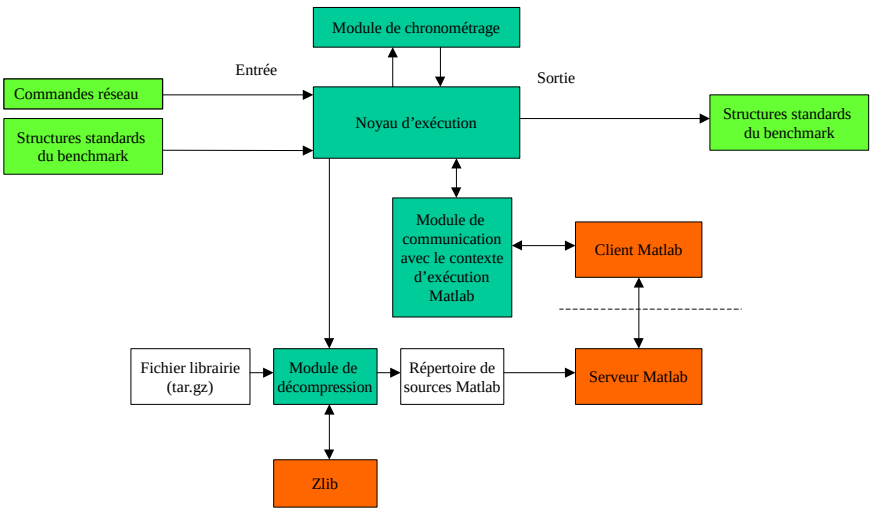
Le serveur Matlab fonctionnant sous linux, il nous parut plus simple d'utiliser le format d'archive tar.gz car il est libre de droit contrairement à d'autre, et que la librairie gzip permettant de développer le décompacteur est fournie en standard sur ce système. Ce format est également très répandu dans la communauté informatique, ce qui conviendra parfaitement à notre philosophie de développement.

Un module de désarchivage de fichier tar est réalisé, puis un module de désarchivage de fichier gz.

Il ne reste plus qu'à écrire un module multi plate-forme de création et d'exploitation de fichiers temporaires pour que notre système de lecture de librairie Matlab soit terminé.

Une fois les librairies Matlab exploitables, c'est à dire temporairement décompactées, il faut à présent les exécuter : il faudra donc communiquer au serveur Matlab. Pour cela nous disposons d'une librairie cliente fournie par Matlab. Une simple encapsulation des appels à cette librairie dans un module de communication sera suffisante pour envoyer simplement les résultats obtenus au serveur de benchmark. Ce module se chargera de la conversion entre le contexte d'exécution Matlab et l'architecture mémoire choisie pour la représentation des données.

Noyau du serveur de librairie Matlab



4.2.2. Déroulement concret des études

Fonctionnalité standard

Le noyau du benchmark sera livré en standard avec trois serveurs de librairie qui offriront la possibilité d'exécuter des librairies Windows (Dynamic Link Library) des librairies NetBSD (Shared Object) et des librairies Matlab (True ARchive Gnu Zip A.S.C.I.I.)

Les fichiers stockés dans la banque d'image seront au format ppm, pgm et jpeg
Ces formats d'image sont libres de droit, le premier gère les images RGB, l'autre gère seulement le niveau de gris, le dernier gère la luminance et le RGB. Les deux premiers formats sont très utilisés dans le milieu de la recherche car leur architecture est très simple : une entête en A.S.C.I.I. suivie de la représentation mémoire directe de l'image.
Leur simplicité d'utilisation a un coût : la taille des fichiers est volumineuse car ce format ne possède pas d'algorithme de compression, c'est pourquoi nous implanteront également le format JPEG, peut coûteux en place, mais détériorant la qualité de l'image en fonction du taux de compression.

Le chargement d'une image ppm ou pgm sera effectué par la même librairie de chargement d'image, car le code de chargement est relativement identique. La seule différence notable est le nombre de bit par pixel, c'est pourquoi un module de conversion des codes de luminance en code RVB et inversement sera écrit de manière à retranscrire le contenu du fichier dans la représentation mémoire standard des média au niveau du benchmark.

Pour la lecture d'une image JPEG, la librairie de chargement sera développée grâce à la libjpeg qui nous allègera le travail, il faudra tout de même reconstruire la structure standard du média en prenant garde au nombre de bit par pixel. Chose facilitée par le module développé précédemment.

Bien sur il serait possible de gérer bien plus de format de librairie et d'image, mais le temps de développement nous est compté. Néanmoins, l'architecture du noyau laisse suffisamment de portes ouvertes pour le faire évoluer à moindre coût.

Test

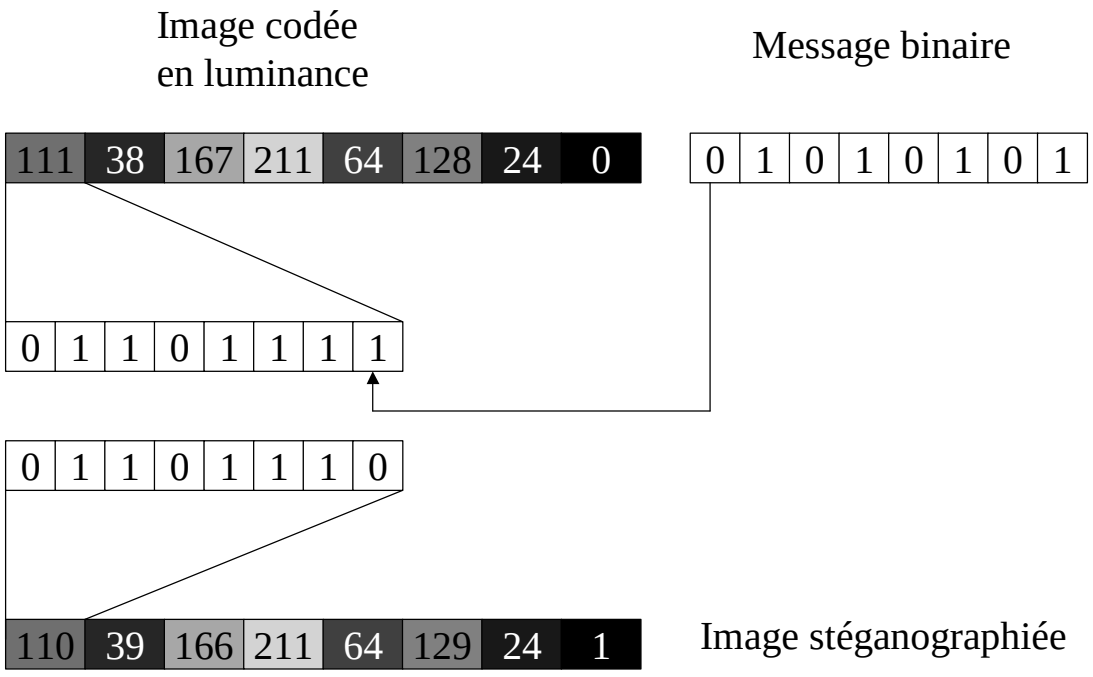
L'ensemble du noyau de benchmark a été bâtit sur une multitude de modules indépendants. Pour chaque module, un petit programme d'appel aux fonctions " publiques " était généré de manière à vérifier le bon fonctionnement des modules du projet. Ces petits programmes ont tous été recompilés et exécutés sur les plates-formes potentiellement utilisées.

Des macro définitions de déboguage ont également été écrites pour insérer un mécanisme d'assertion et de traçage conditionnel de certaines fonctions et variables. Ce système à permit de tracer le code à moindre coup sans pour autant surcharger le binaire exécutable.

Pour tester le fonctionnement global du noyau, des librairies utilisateurs ont du être écrites.

En effet nous avons écrit une librairie de watermark élémentaire pour tester le benchmark : il s'agit d'un algorithme de stéganographie non robuste. Il se contente de stocker les bits de la donnée à watermarker dans les bits les moins significatifs des composantes rouge, vert et bleue de l'image. Nous utiliserons cette méthode pour insérer la taille en binaire naturel de la donnée dans l'image afin de la retrouver lors du décodage.

Steganographie élémentaire



Pour vérifier que l'image est correctement chargée et correctement watermarkée, un scénario sera écrit afin de tester si la relecture de la marque se fait correctement.

Un writer d'image ppm sera écrit afin de visualiser à tout moment les médias contenus dans le contexte d'exécution. Ceci nous permettra de constater la qualité visuelle de l'algorithme de watermarking.

Une attaque sera également développée pour tester les algorithmes de watermarking robustes. Nous avons choisis un simple flou ou l'on peut paramétrer la taille du rectangle de moyenne, ceci permettra donc de vérifier que le système d'argument optionnel fonctionne correctement.

Le writer de ppm permettra de visualiser l'image attaquée, on constatera grâce au scénario de test que la marque stéganographiée n'est plus lisible.

Une librairie de test de qualité sera également développée pour évaluer les algorithmes. Nous utiliserons la méthode du PSNR (Peak Signal to Noise Ratio) basée sur le calcul du MSE (Mean Square Error) cette méthode renverra un coefficient qui nous permettra de connaître l'indice de qualité qu'il y a entre l'image de base et l'image modifiée.

Nous remercions également Frédérique Lefebvre pour les efforts qu'il a fournis, notamment en tant qu'auteur des algorithmes de test pour la plate-forme Matlab. Mais également pour le transfert de compétence effectué nécessaire à l'élaboration d'un serveur sur cette plate-forme.

4.3. Interprétation et critique des résultats

Malgré tous les efforts effectués pour rendre le benchmark le plus modulaire possible, son utilisation sur certains types de média risquent d'être compromise à cause de son architecture réseau.

En effet, traiter des médias comme le son ou la vidéo serait quasi-impossible car les structures génériques correspondantes à ces médias sont d'une taille phénoménale. Il est pour le moment peut probable de pouvoir faire circuler de telle structure sur le réseau à un débit respectable.

Le benchmark est plus adapté à des médias comme les images, les objets filaires, les documents textes…

La plus grosse difficulté rencontrée fut le portage du réseau coté serveur sur l'architecture Win32. Celle-ci permettait d'avoir un code source commun aux versions Linux et Unix. Malgré le portage du système de socket Berkeley effectué par Microsoft dans le système Winsock 2, Il existe de gros problèmes de compatibilité et des comportements suspects par rapport aux RFC établis dans la version BSD. L'équipe de développement du logiciel Apache a apparemment rencontré les mêmes problèmes.

5. Conclusion générale

L'élaboration du benchmark terminée, nous mettons à disposition de la communauté scientifique un outil de test qui simplifiera les futurs développements en matière de watermarking d'image.

L'écriture modulaire du benchmark permettra au développeur de porter le benchmark pour d'autres types de média.

On pourra également écrire des serveurs de librairie pour d'autres plates-formes et d'autres types de librairie sans même modifier le noyau de l'application.

Le grand public pourra également compléter le benchmark en chargeant de nouvelles librairies.

L'optique de développement modulaire et ouvert au futur, permettra au benchmark de survivre aux changements de technologie et aux extensions des domaines d'utilisation du watermarking.

Cette approche était nécessaire pour le développement d'outils liés à des technologies émergentes.

Le développement au sein du monde de la recherche nous a permis d'avoir une approche plus ouverte sur l'évolution des projets, le travail indirect avec d'autres chercheurs, la contribution et l'échange de solution technique pour la communauté scientifique.

Ceci nous éloigne énormément de l'obscurantisme volontaire des progiciels propriétaires destinés à la vente. Ce qui permet à chacun de rester indépendant, autonome et concentré sur ses recherches.

Le monde scientifique n'est pas le seul bénéficiaire de cette outil, car grâce à lui, la vulgarisation de l'utilisation du watermarking dans tous les domaines et les choix relatifs aux technologies cibles en seront facilités.

6. Bibliographie & glossaire

Scanning the Special Issue on Identification and Protection of Multimedia Information, *B. Macq* in Proceedings of the IEEE July 1999 Vol. 87 No. 7, pp. 1059 – 1061

Information Hiding – A Survey, *F.A.P. Petitcolas, R.J. Anderson, and M.G.Kuhn* in Proceedings of the IEEE July 1999 Vol. 87 No. 7, pp. 1059 – 1077

The Use of Watermarks in the protection of Digital Multimedia Products, *G. Voyatzis and I. Pitas* in Proceedings of the IEEE July 1999 Vol. 87 No. 7, pp. 1197 – 1207

Secure Delivery of Images over Open Networks, *D. Augot, J.-M. Boucqeau, J.-F. Delaigle, C. Fontaine, and E. Goray* in Proceedings of the IEEE July 1999 Vol. 87 No. 7, pp. 1251 – 1266

Nasa Applied Information Technologie Division WEB site, Peak Signal to Noise Ratio and Mean Square Error Documentation http://appliedit.arc.nasa.gov/videotech/psnr.html http://appliedit.arc.nasa.gov/videotech/quality.html

7. Annexes

7.1. Sommaire des annexes

1- Documentation technique du benchmark Page 34

2- Fonctionnalités des Benchmarks existants Page 56