Programmation

# Compression de données ...

**Christian Quest**
**cquest@cquest.org**

# Qui suis-je ?

- Auteurs de plugins 4D
  - Internet-ToolKit (ITK)
  - Stuff-ToolKit
  - Serial-ToolKit (STK)
  - PGP-ToolKit (expérimental)
- Développements pour ACI
  - Premiers composants réseaux de 4D Server (ADSP et TCP/IP)
  - 4D Remote
- Administrateur de 4d-forum, membre du comité de rédaction de Planète4D

# But de cet atelier

- Découvrir ce qu'est la compression de données
- Les atouts de la compression de données
- Le principe général
- Les différents types de compression
- Les différents algorithmes et outils
- Compression de données dans 4D

# La compression de données

- Réduire le volume de données

# Les atouts

- Gain en espace de stockage
  - Sur disque
  - En mémoire
- Gain en temps sur les lectures/écritures
  - Moins de données à lire/écrire physiquement sur disque
  - Compression + écriture plus rapide qu ’une écriture des données brutes
  - Lecture + décompression plus rapide qu ’une lecture des données brutes
- Gain sur les temps de transmission
  - Moins de données à transmettre
  - Compression + envoi / réception + décompression plus rapide

# Les atouts : exemples

- Gain en espace de stockage
  - Une image JPEG se compresse facilement à 90%
  - Un fichier HTML compressé en GZIP se comprime facilement à 50% de sa taille originale

# Les atouts : exemples

- Gain en temps sur les lectures/écritures
  - Temps pour écrire une image de 14Mo: 5s (soit 2,8Mo/s)
  - Temps pour compresser une image de 14Mo en JPEG: 3,4s, soit 6,5Mo/s, résultat: 503Ko (soit 4% de l 'original)
  - Temps pour écrire une image JPEG de 503Ko: 0,2s
  - Temps total compression + écriture: 3,6s, soit 1,4s de moins (gain de 24% en temps et 96% en volume stocké sur disque).

# Les atouts : exemples

- Gain sur les temps de transmission
  - Temps de transmission à 56Kbps/s d'un fichier HTML de 30Ko: 5,4s
  - Temps de compression du fichier HTML au format « GZIP »: 0,05s (soit 600Ko/s), taille compressée: 15Ko (gain de 50% en volume)
  - Temps de transmission de l'HTML compressé: 2,7s
  - Temps total compression/transmission/décompression: 2,8s, au lieu de 5,4s, soit un gain de près de 50% en temps

# Autres atouts

- Les formats d'archive (comme ZIP ou Stuffit) réduisent le nombre de fichiers
  - Un seul fichier ZIP ou Stuffit peut contenir des centaines voire des milliers de fichier
  - Ces fichiers peuvent être organisés hiérarchiquement
- Les fichiers compressés sont «vérifiables» (notion de checksum)

# Les différents types de compression

- Les compressions « textuelles »
  - Les données décompressées sont exactement les même que les données originales
  - On peut compresser/décompresser les données autant de fois que l ’on veut
  - Exemple: fichiers Stuffit, ZIP
- Les compressions destructrices
  - Les données décompressées ne sont pas exactement les même que les données originales
  - Les compressions/décompressions successives détruisent petit à petit les données

# Les compressions « textuelles »

- Supprimer la notion d'octets, voire même de bits
- Rechercher et éliminer les répétitions

# Le principe de base

- Comment mettre plus de pommes de terre dans des boites ?

- En faisant de la purée !

# Supprimer la notion d'octets ou de bits

- Un octet contient 8 bits, mais souvent on ne se sert que de 7bits, voire même de moins.
- Exemple:
  - un code barre est composé d'une douzaine de chiffres (12 octets en texte ASCII),
  - chaque chiffre a une valeur de 0 à 9, soit moins de 4 bits utiles (exactement log(10)/log(2) soit 3,322 bits),
  - En codant chaque chiffre sur 4 bits, on gagne déjà 50% en volume (6 octets au lieu de 12)
  - En codant exactement sur log(10)/log(2) bits, on gagne plus de 58% (5 octets au lieu de 12)

# Rechercher les répétitions

- Certaines informations sont plus souvent présente que d'autres dans les données que l'on veut compresser.
- Exemple: dans un fichier HTML, on trouvera beaucoup de signes < , / , et >
- On va chercher à coder les données se répétant souvent sur moins de bits, et les données moins fréquentes sur plus de bits.
- Principe de la compression « Huffman »

# Eliminer les répétitions

- On va chercher à élargir la réduction des répétitions à des groupes d'octets plutôt que des octets simples.
- Exemple: dans un fichier HTML, on trouvera souvent des balises comme <B> </B>, que l'on cherchera à coder comme en bloc.
- Principe utilisé par les compressions Lempel-Ziv, zip, gzip

# Autres principes

- Le codage arithmétique (très efficace) utilise une notion de probabilité
- il casse la barrière du bit

# Le principe général de la compression

- Un modèle fourni une probabilité d'apparition d'une information
- On code l'apparition d'une information à partir du modèle
- Le modèle peut être fixe, ou s'adapter aux données au fûr et à mesure de la compression

# Le principe général de la décompression

- Le même modèle fournit une probabilité d'apparition des information
- On décode l'apparition d'une information à l'aide du modèle
- On met à jour le modèle si on est dans un mode "adaptatif"

# Exemple: compression "Huffman"

- Message à coder: abracadabra
- Fréquence de chaque caractères: a=5, b=2, c=1, d=1, r=2
- Résultat codé: 23 bits soit 73,8% de gain
- Inconvénient: besoin de transmettre la table des fréquences (le modèle)

# Compression adaptative

- Elle adapte le modèle automatiquement aux données
- Généralement plus efficace
- Pas besoin de transmettre le modèle, on commence avec un modèle "de départ"
- Peut fonctionner "à la volée"

# Compression non adaptative

- Nécessite une analyse préalable des données, ou bien l'utilisation d'un modèle fixe
- Peu utilisée car généralement moins performante
- Utilisée par exemple par les Fax (compression de Huffman avec une table fixe de fréquence)

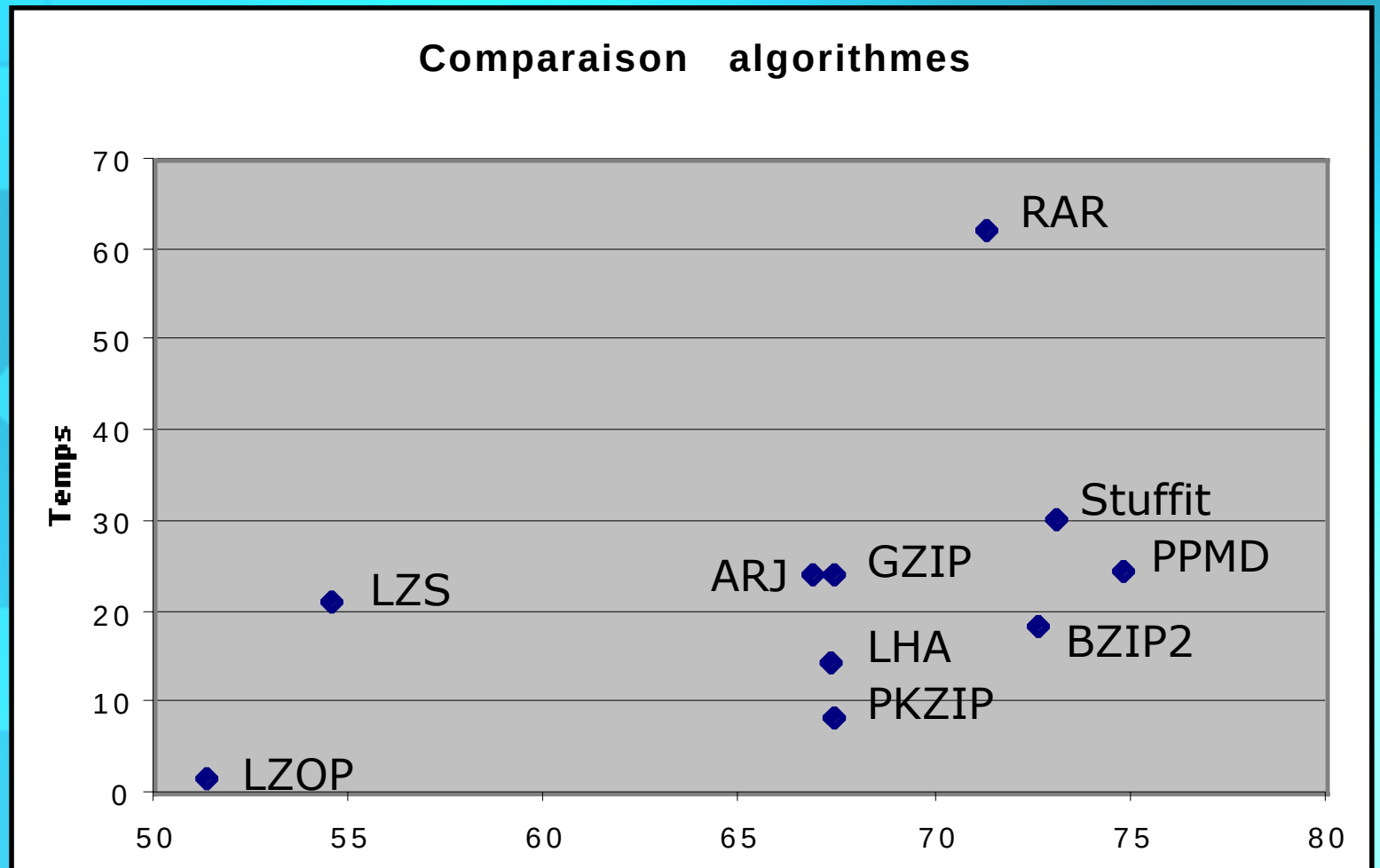
# Les algorithmes de compression « textuelle »

- Les algorithmes statistiques
  - Ces algorithmes se basent essentiellement la fréquence d'apparition des données (Huffman, Codage Arithmétique)
- Les algorithmes à base de dictionnaires
  - Ces algorithmes ne nécessitent pas de passe initialise d'analyse. Ils sont les plus répandus.

# Les algorithmes de compression « textuelle »

- Huffman: 1952, peu efficace
- Lempel-Ziv
  - LZ77: 1977 - 3,94 bpc (Bits par caractère)
  - LZMW: 1984 - 3,32 bpc
  - LZH: 1987 - 3,30 bpc
  - LZB: 1987 - 3,18 bpc
- MTF (Moffat): 1987 - 3,24 bpc
- GZIP: 1987 - 2,71 bpc
- PPMC (Moffat): 1988 - 2,48 bpc, puis 2,34 (1993)
- PPMD (Teahan): 1995 - 2,27 bpc

# Comparaison des algorithmes



# Conclusion de cette comparaison

- Certains algorithmes (PPMD, Stuffit, ARC, RAR) sont les plus efficaces (plus de 70% de gain)
- Une famille d'algorithmes "ZIP" très proches au niveau gain, mais plus ou moins rapides (implémentations différentes)
- Certains algorithmes sont extrèmement rapide (LZOP) mais compriment moins bien

# Autres particularités

- Certains algorithmes sont optimisés en décompression (décompression très rapide, mais compression bien plus lente)
- Certains algorithmes sont optimisé pour leur économie de mémoire

# Les compressions destructrices

- Elles se basent sur la perception que l'on a de l'information et vont éliminer les informations que nous ne pouvons pas percevoir
  - Exemple: avec JPEG, les informations de couleur sont codées avec moins de précision que les informations de luminosité car l'œil humain est moins sensible aux couleurs qu'à la luminosité

# La compression GIF

- Ce n'est pas une compression destructrice… mais…
- Elle est limitée à une palette de 256 couleurs (forme de destruction d'information pour les images originales contenant plus de 256 couleurs)
- Inadaptée aux images "photographiques"

# La compression JPEG

- JPEG = Joint Photographic Expert Group
- Elle est adaptée à la compression d'image « photographiques », c'est à dire en ton continus.
- Le principe :
  - L'image est découpée en carrés de 8x8 pixels.
  - Pour chaque carré, une luminosité et une couleur moyenne sont calculées, puis la variation de luminosité et de couleur de chaque pixel est codée en zig-zag en partant du coin haut-gauche.

# La compression JPEG

- Avantages
  - Pas de notion de palette de couleurs, c'est un codage "fréquentiel" (analyse des fréquences
  - Code "progressif" possible (on encode en plusieurspasse, chaque passe étant de plus en plus détaillée)
  - Permet de choisir un rapport compression/qualité

# La compression JPEG

- Inconvénients
  - Le découpage en carrés de 8x8 pixels fait rapidement apparaître des défauts à la frontière de ces carrés
  - Inadapté aux images non "photographiques" (graphiques)
  - Pas de gestion de transparence (canal alpha par exemple)

# La compression JPEG

- JPEG 2000: l'évolution de JPEG
- Basé sur la compression par ondelettes
- Plus de découpage en 8x8 pixels
- Meilleure compression et de meilleure qualité
- En cours de standardisation
- Format à "options"

# La compression MPEG

- Elle est adaptée à la compression de séquence d'images (films, vidéo)
- Le principe :
  - Chaque image est compressée avec un système proche du JPEG (découpage en carrés)
  - A cela s 'ajoute une détection des mouvement et des parties fixes de l 'image
- Utilisée par les DVD, les chaînes numériques, certains appareil photo numériques, voire caméscopes/magnétoscopes numériques

# La compression MPEG

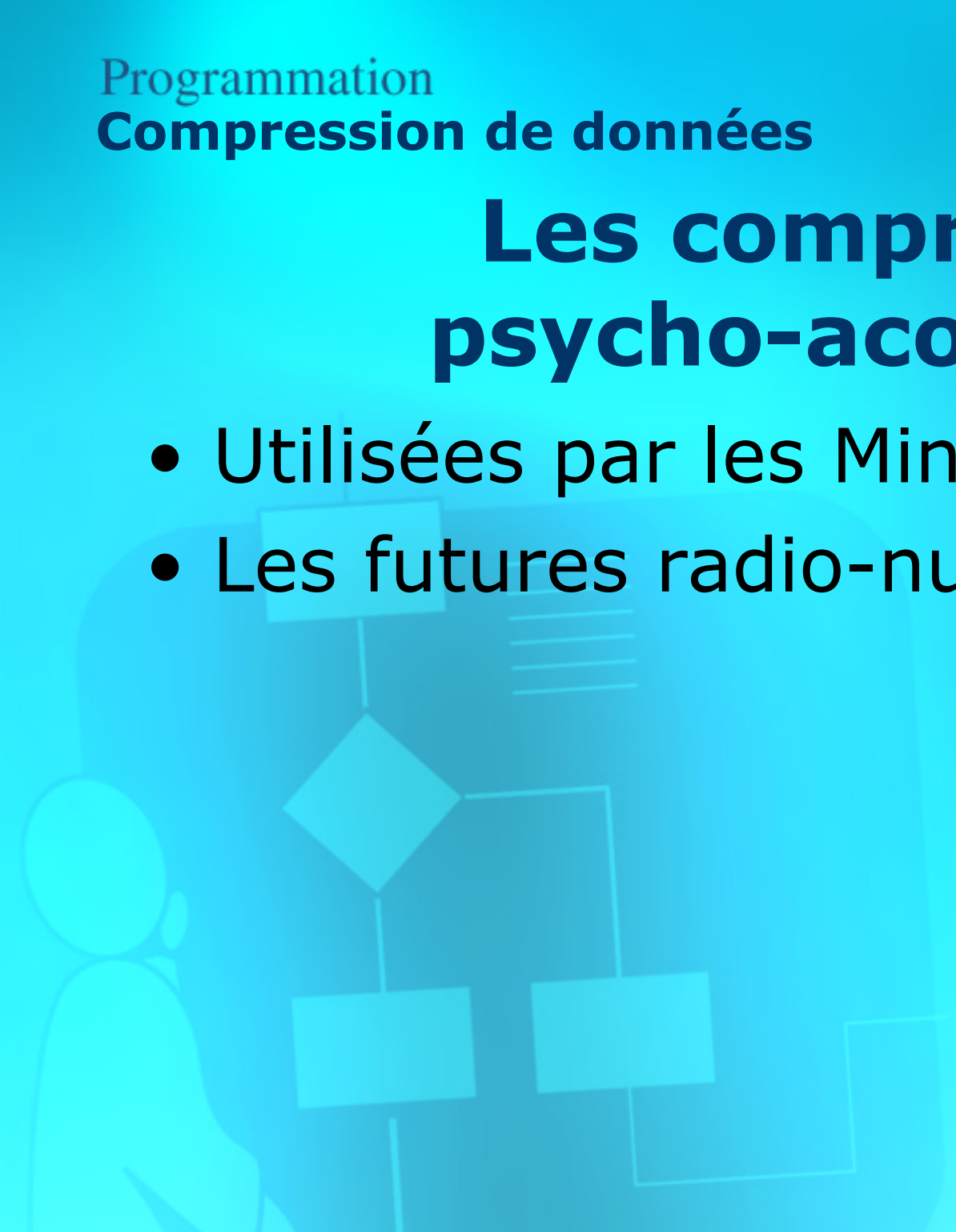
- Comme JPEG, on peut choisir le rapport compression/qualité
- Même genre de défauts liés au découpages des images en blocs de 8x8 pixels

# La compression MP3

- Elle est adaptée à la compression de sons de musique
- C'est un sous éléments de la norme MPEG
- Le principe :
  - L'oreille humaine ne perçoit qu'environ 10% de l'information sonore du fait de limitations mécaniques
  - On parle de compression psycho-acoustique
  - Les signaux de fréquences voisines se masquent, un signal fort masque les signaux proches plus faibles, etc.

# Les compressions psycho-acoustiques

- Utilisées par les MiniDisc
- Les futures radio-numériques (DAB)



# Compression de données dans 4D

- Compression "textuelle" de fichiers/texte/données
- Compression "destructrices" d'images/sons/vidéo

# Les outils de compression pour 4D

- COMPRESSER BLOB
  - Compression "textuelle" propriétaire à 4D (basée sur LZS)
- Utilisation de plugins
  - Stuff-ToolKit
  - Qpix
  - Qmedia

# Stuff-ToolKit
# Stuffit-Engine dans 4D

- Permet d'exploiter les fonctionnalités de Stuffit-Engine depuis le langage de 4D
  - Compression/Décompression Stuffit, ZIP, BZIP
  - Décompression d'une multitude d'autres formats/algorithmes
  - Support des "images disque"
  - Disponible pour MacOS (version Windows prévue)
  - Version "Pro" intégrant Stuffit-Engine
- DEMO

# Qpix (Escape)
# Compression d'images

- Permet d'exploiter les fonctionnalités de compression/décompression d'images de QuickTime
- Support des (très) nombreux formats d'images reconnus par QuickTime
- Disponible pour MacOS et Windows

# QMedia (Escape)
# Compression de "moovies"

- Permet d'exploiter les fonctionnalités de compression/décompression de "moovies" de QuickTime
- Support des (très) nombreux formats de vidéo reconnus par QuickTime
- Disponible pour MacOS et Windows

# Questions/Réponses