De nouveaux algorithmes de compression pour les fichiers standards

Yves Chartier
Services informatiques, DRM
Février 2005

(mise à jour : Juin 2006)

Plan de la présentation

- La nature du problème
- Entropie informationnelle
- Algorithmes implantés
  - Méthode minimum
  - Méthode bicubique
  - Méthode de Lorenzo
- Quelques statistiques
- Conclusions

La nature du problème

- Notre puissance de calcul augmente plus vite que notre capacité de stockage

Extrait d'un courriel de Richard Hogue
“Implantation du nouveau modèle GEM-15km mardi prochain 18 mai 2004”

- *“La taille des fichiers du GEM-15km (576x641) est 2.5 fois plus grande que les fichiers du GEM-24km (354x415). Notez qu'en coordonnée eta le facteur est près de 5 fois plus grand puisqu'il y a 30 niveaux de plus avec le GEM-15km (58 niveaux vs 28). À titre d'exemple, à chacune des passes du nouveau modèle régional GEM 15km les sorties en coordonnée pression et eta produiront 29Gb par passes (vs 6Gb actuellement).*
- *L'ensemble du volume de données relié au système régional (anal, trial, prog) passera de **17Gb à 70Gb** par jour avec l'implantation du nouveau système.”*

Et ça continue...

- On est rendus (mai 2006) à 120 Gb par jour (70 en 2004)
- Une run du méso-global produit 40 Gb de données.

Souvenirs de conférence...

- Vizualisation 2003, Seattle...
- Dans un séminaire portant sur la compression de données, l'implantation d'un algorithme prédicteur-correcteur a démontré comment une simulation 4D de 43 Megs pouvait être compressée sans perte à 1.2 Megs (2.5 % de l'original).
- Un cas un peu idéalisé (données à 8 bits et avec beaucoup de redondance) mais montre cependant le potentiel de la méthode.

Compression de base

- Il existe déjà une compression de base dans les fichiers standards...
- Donné par le paramètre NBITS
  - Valeurs typiques... 24, 16, 12, 8
  - NBITS = 16 donne un ratio de compression de 2
  - NBITS = 12 donne un ratio de compression de 2.5
  - NBITS = 8 donne un ratio de compression de 4...

Quelques essais avec les utilitaires conventionnels

Here are a few Compression ratios obtained :

	bzip2 -9	gzip -9	compress
Uncompressed model run (dynamics)	1.17 : 1	1.17 : 1	No compression
Uncompressed model run (physics)	1.50 : 1	1.49 : 1	1.08 : 1
12-bit compress Regional GEM	1.46 : 1	1.16 : 1	No compression
Radar data	27.85 : 1	22.09 : 1	20.43 : 1

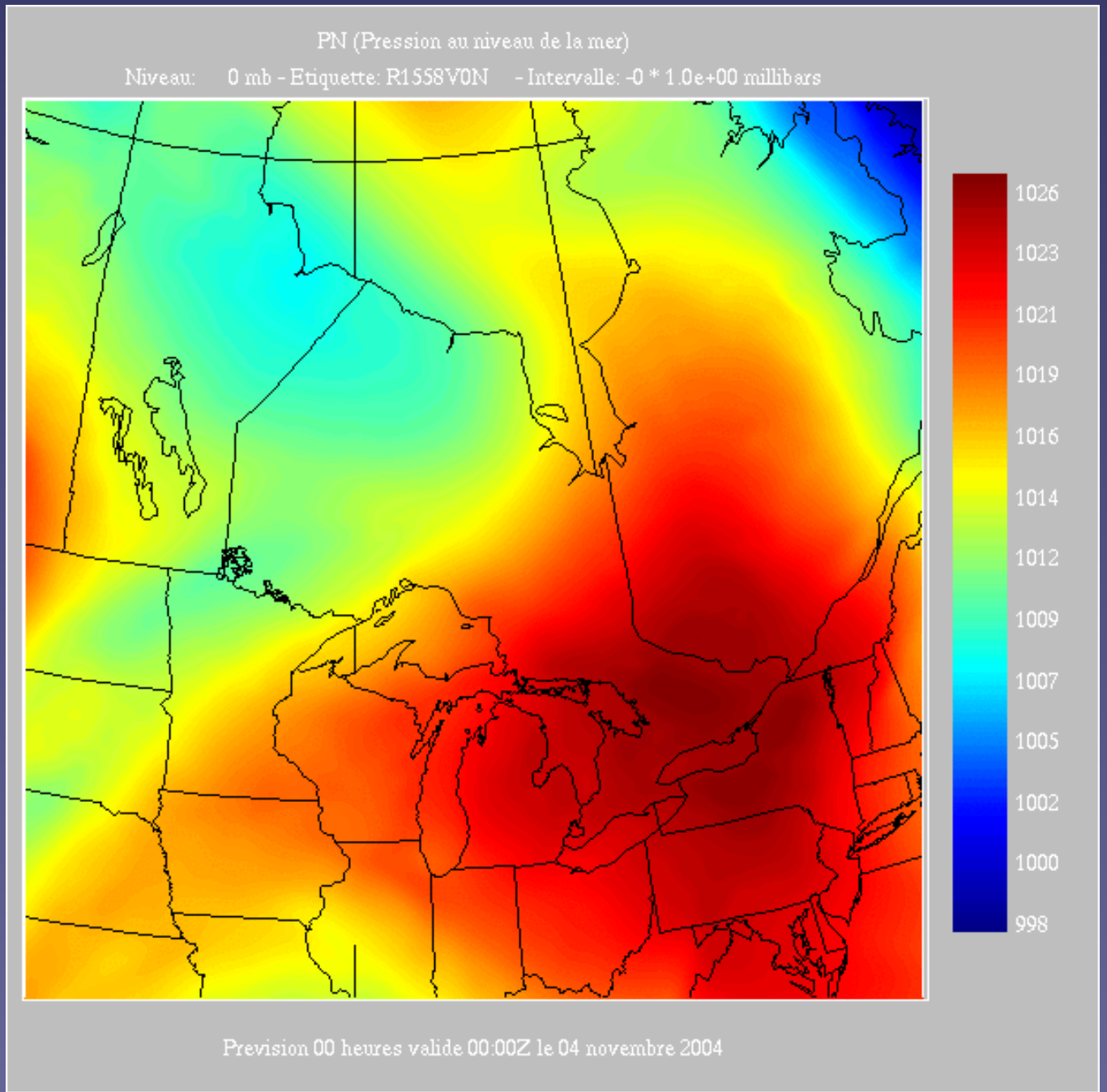
Quantification des données à l'intérieur des fichiers standards

- La formule ci-dessus donne la méthode de quantification de réel à entier utilisée dans les fichiers standards.

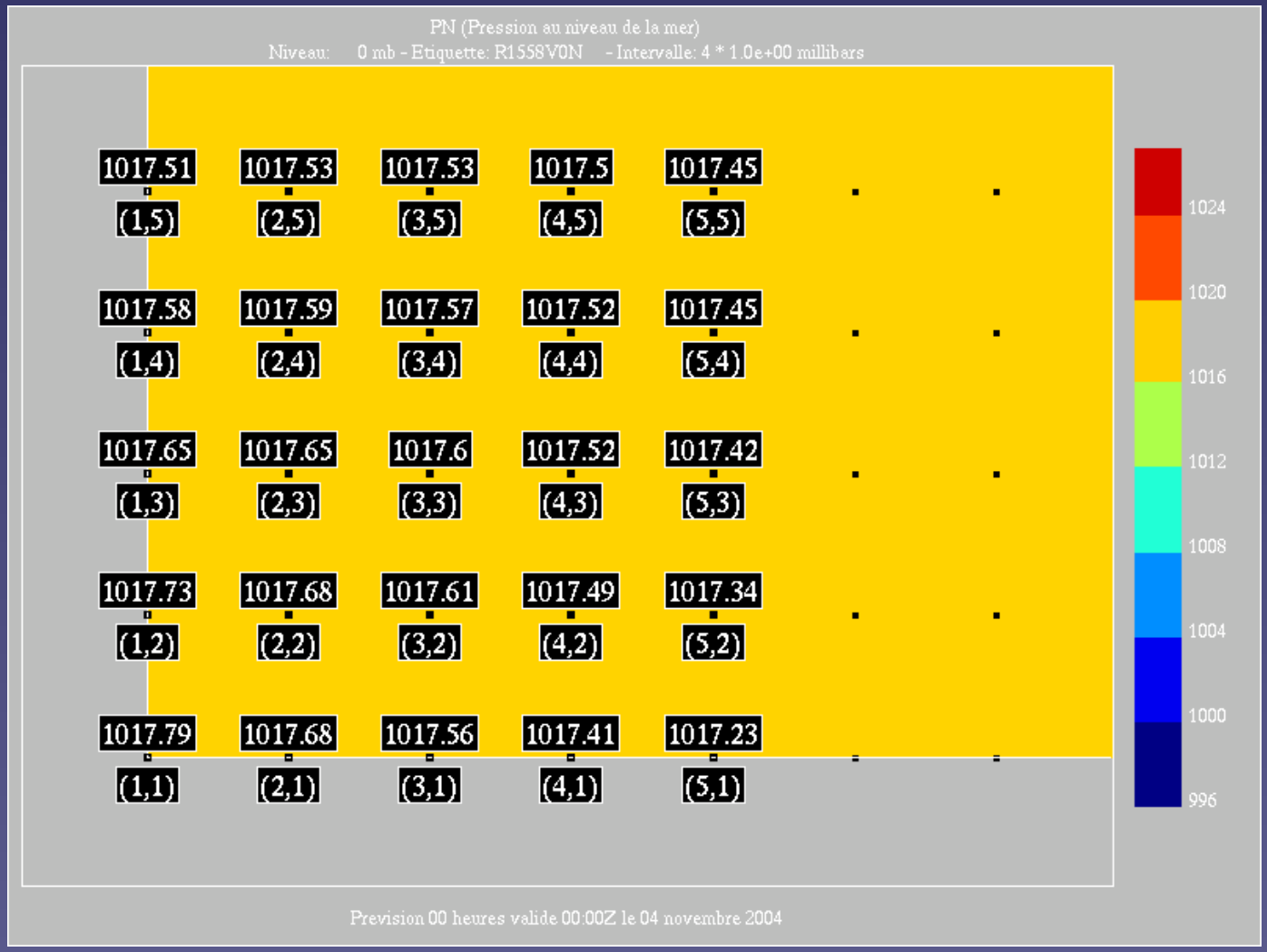
$$INT = 2^{nbits} * (REAL - min) / 2^{N}$$

  - Le dénominateur est 2^N, N étant l'exposant le plus proche de sorte que $2^N > (max - min)$

Un exemple de calcul



Un exemple de calcul



- Max = 1026.71301
- Min = 997.797913
- Delta = 28.9151
- Exposant normalisé = 32
- Nbits = 16
- fld(1,1) = 1017.78619
- ifld(1,1) = 2^{16} * (1017.78619 – 997.79791) / 32.0 = 40936

Quantification

1017.58	1017.59	1017.57	1017.52	1017.45
1017.65	1017.65	1017.60	1017.52	1017.42
1017.73	1017.68	1017.61	1017.49	1017.34
1017.79	1017.68	1017.56	1017.41	1017.23

40373	40415	40417	40340	40254
40515	40537	40498	40389	40240
40665	40659	40551	40659	40551
40812	40727	40565	40331	40565
40936	40726	40474	40166	39804

(

Que sont mes zéros devenus ?

- Soit une analyse de précipitations écrite en R12, avec -99.0 comme code de valeur manquante
  - Min = -99.0
  - Max = 64.3
  - Max – Min = 163.3
  - N = 8 (2^N = 256), nbits = 12
  - Dans cet intervalle, 0.0 tombe au point 1583.61
  - Écrit comme 1584 dans le fichier standard
  - À la lecture, 1584 = 0.02418

)

La nature du problème

- Étant donné un ensemble de points ordonnés de dimension ni * nj (* nk), dont la valeur va de 0 à 2^{nbits} -1...
  - a) existe-t-il une méthode pour compresser cet ensemble sans perte de précision supplémentaire ?
  - b) et ce dans un temps raisonnable ?

Potentiel de compression

- a) existe-t-il une méthode pour compresser cet ensemble sans perte de précision
  - Oui. En fait il existe beaucoup de méthodes!
- Le potentiel de compression peut-être défini par **l'entropie** de l'ensemble de points

Définition formelle de l'entropie

If we have a set of independent events Ai, which are sets of outcomes of some experiment S, such that

$$\cup A_i = S$$

where S is the sample space, then the average self-information associated with the random experiment is given by

$$H = \sum P(A_i) i(A_i) = -\sum P(A_i) \log_b P(A_i)$$

where P(A_i) is the probability of occurrence of symbol A_i.

Quelques observations (1)

- Soit un ensemble de 65536 événements (2^{16})
- Il y a seulement 656 événements distincts dans cet ensemble : Le premier se produit 99 fois sur 100, les autres se produisent seulement 1 fois. L'entropie de cet ensemble est

$$-(1\times((\frac{99}{100})log2(\frac{99}{100}))+655\times((\frac{1}{65536})log2(\frac{1}{65536})))=0.17$$

- Le tout tient en 11142 bits au lieu de 1048576
- Ratio de compression = 94

Quelques observations (2)

- Soit un ensemble de 65536 événements (2^{16})
- Il y a 6555 événements distincts dans un ensemble : Le premier se produit 90 % du temps, les autres se produisent seulement 1 fois. L'entropie de cet ensemble est

$$-(1\times((\frac{9}{10})log2(\frac{9}{10}))+6554\times((\frac{1}{65536})log2(\frac{1}{65536})))=1.74$$

- Le tout tient en 114033 bits au lieu de 1048576
- Ratio de compression = 9.2

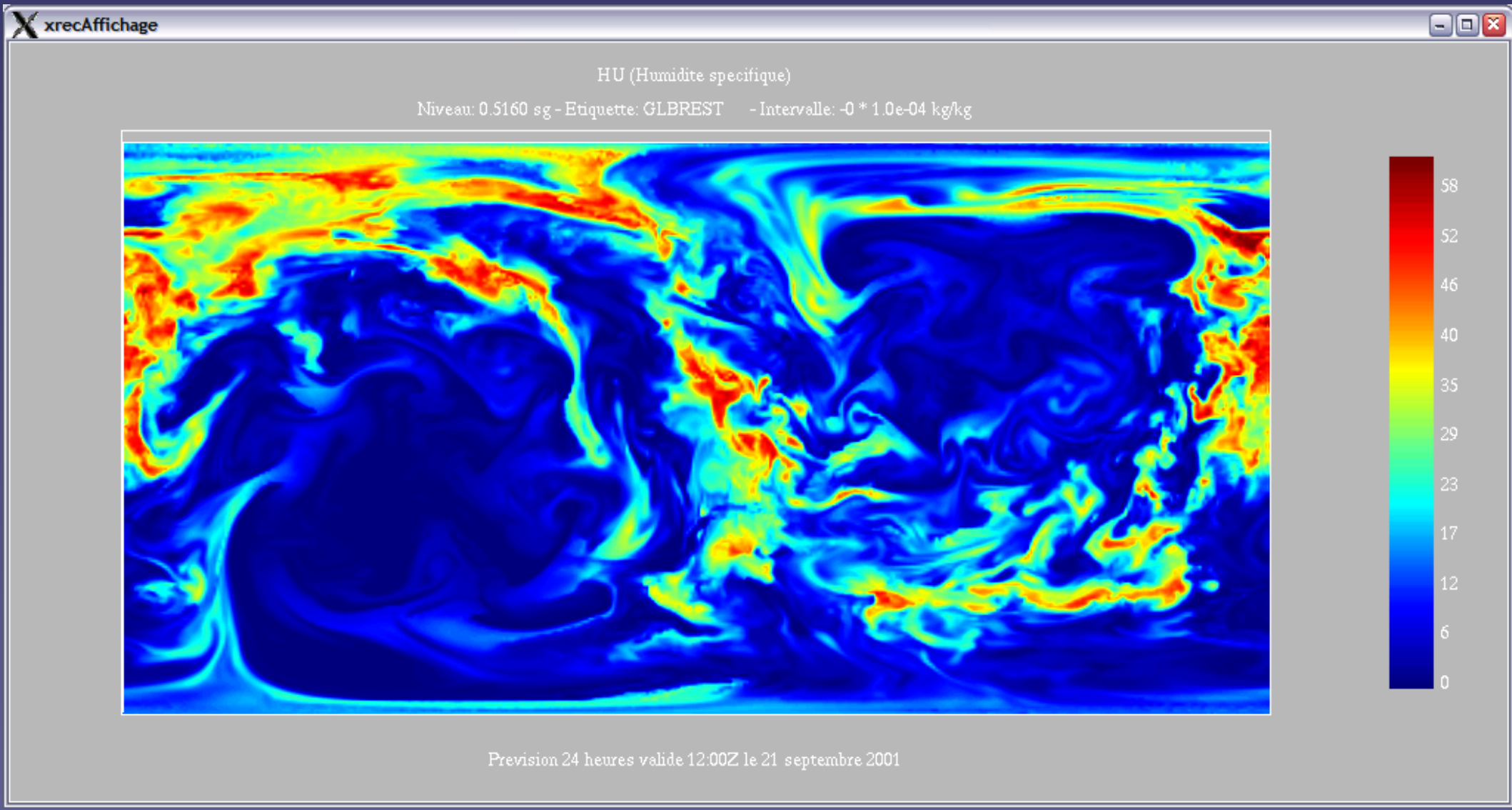
Quelques observations (3)

- Sur un ensemble de 65536 événements
- 25 % se produisent chacun 2 fois, les 50 % qui restent se produisent 1 seule fois. L'entropie de cet ensemble est

$$-(16384\times((\frac{2}{65536})log2(\frac{2}{65536}))+32768\times((\frac{1}{65536})log2(\frac{1}{65536})))=7.75$$

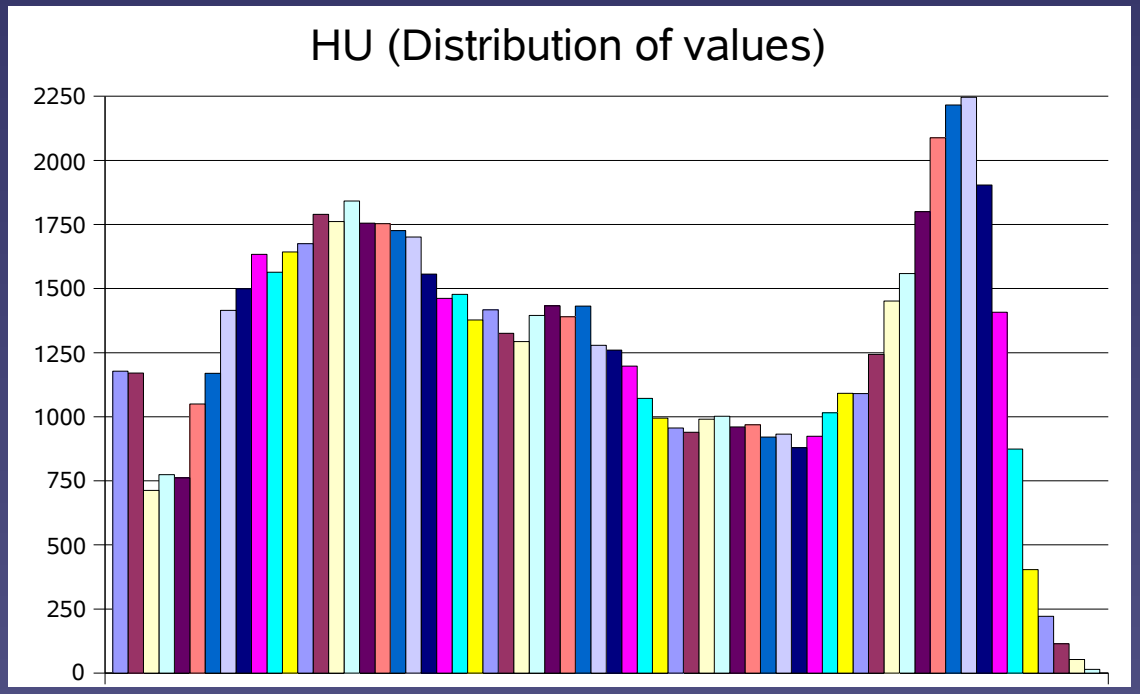
- Le tout tient en 507904 bits au lieu de 1048576
- Ratio de compression = 2.06

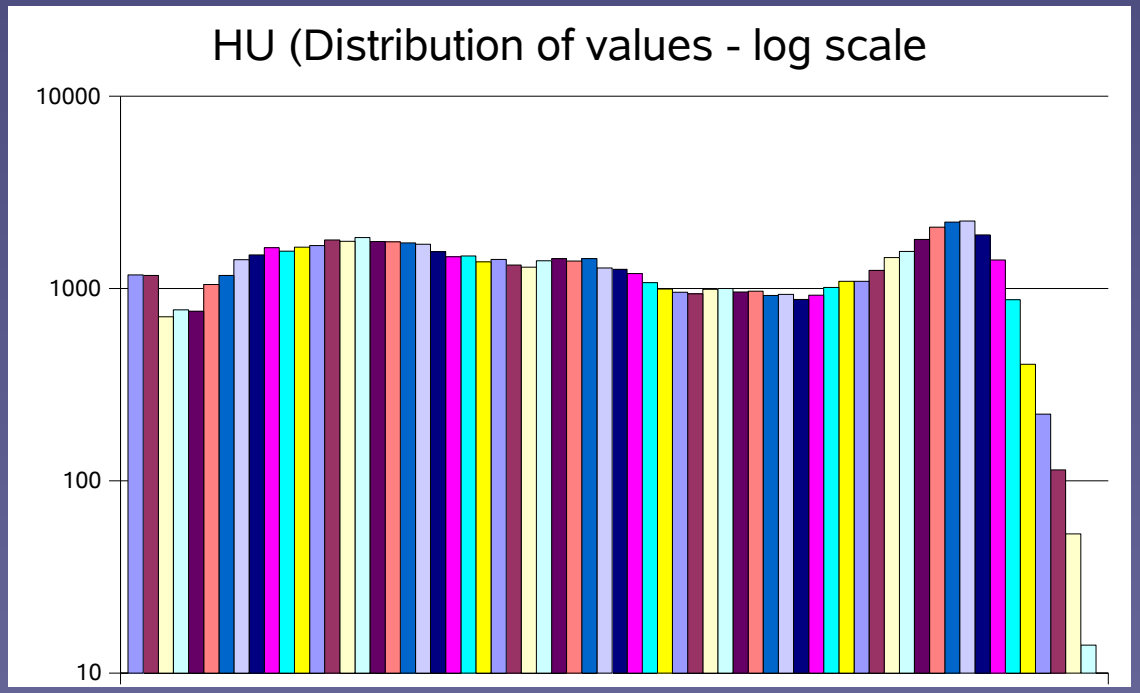
Entropie de quelques champs météo



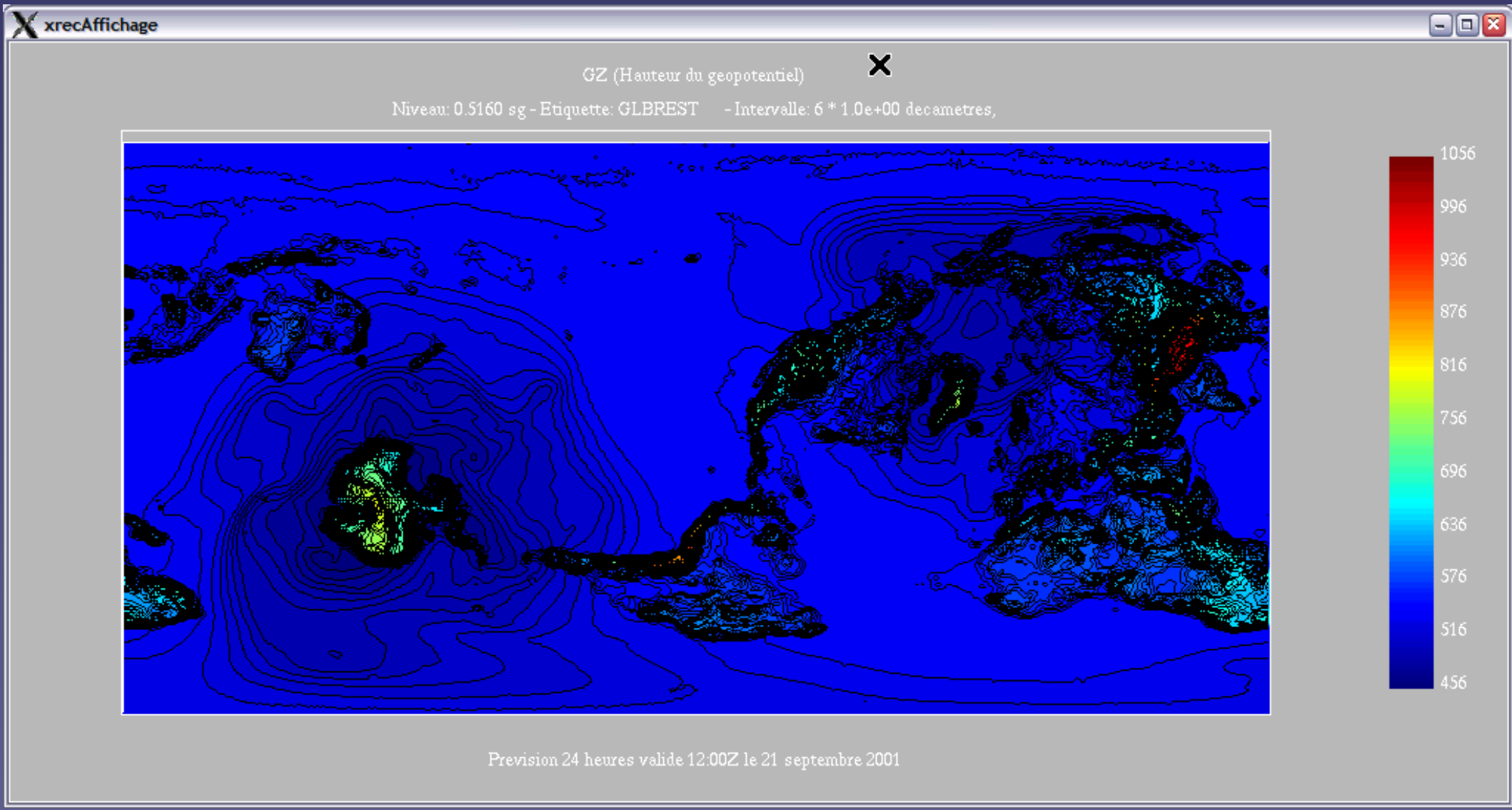
Prog de 24 hrs HU eta = 0.5160 – Entropie : 14.45

Distribution des valeurs (en tranches de 64)



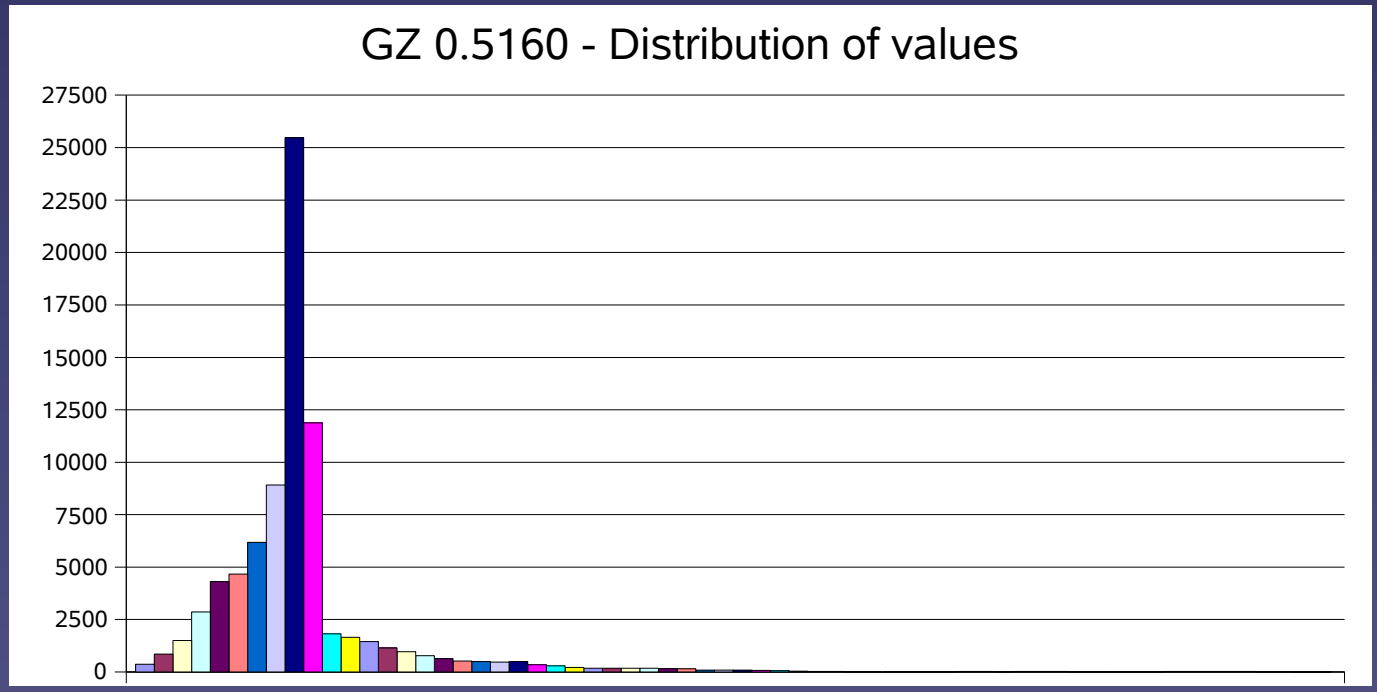


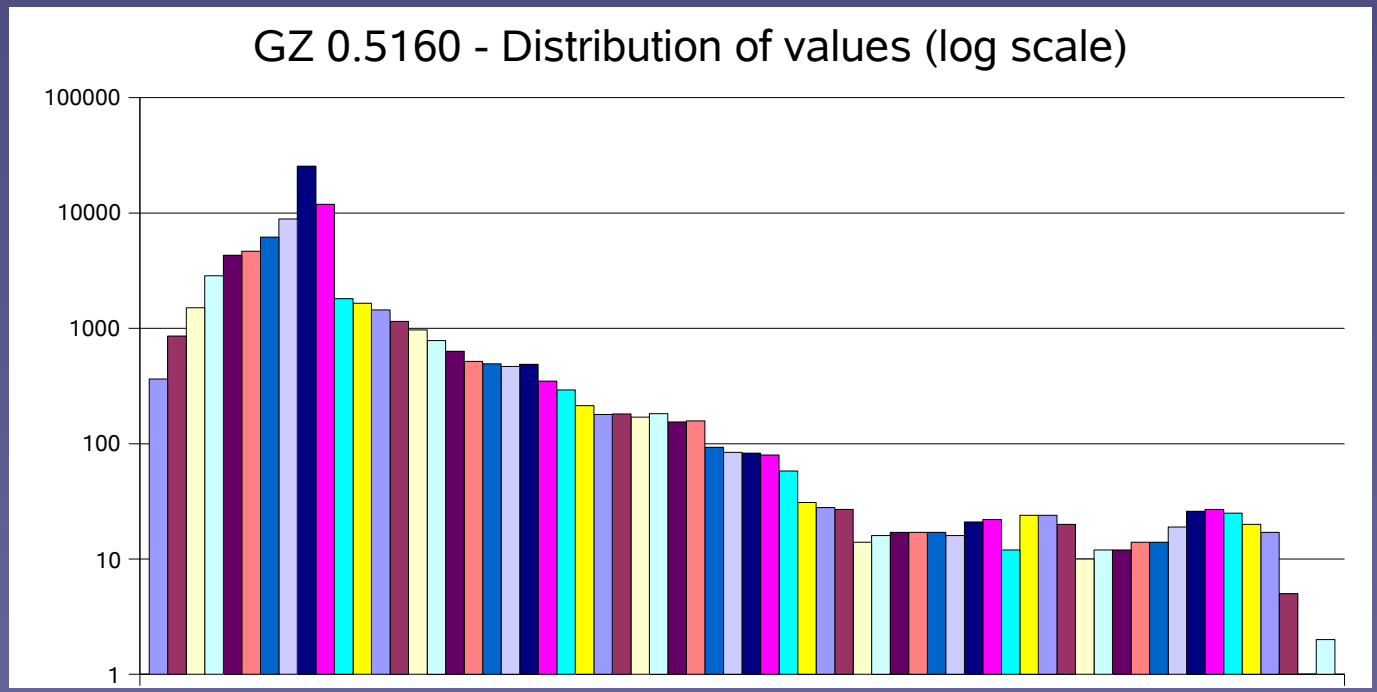
Entropie de quelques champs météo



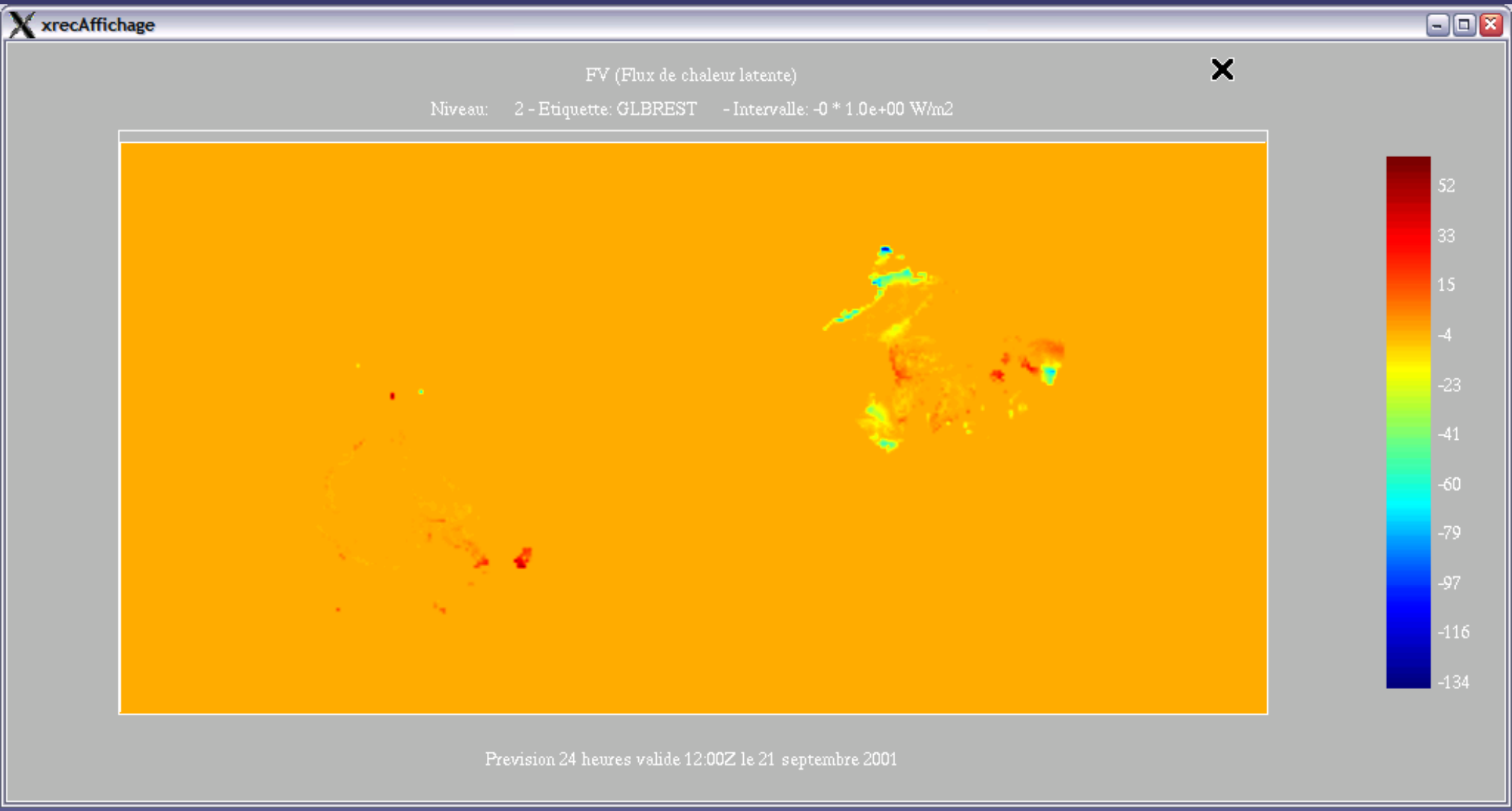
Prog de 24 hrs GZ eta = 0.5160 – Entropie : 12.23

Distribution des valeurs (en tranches de 64)



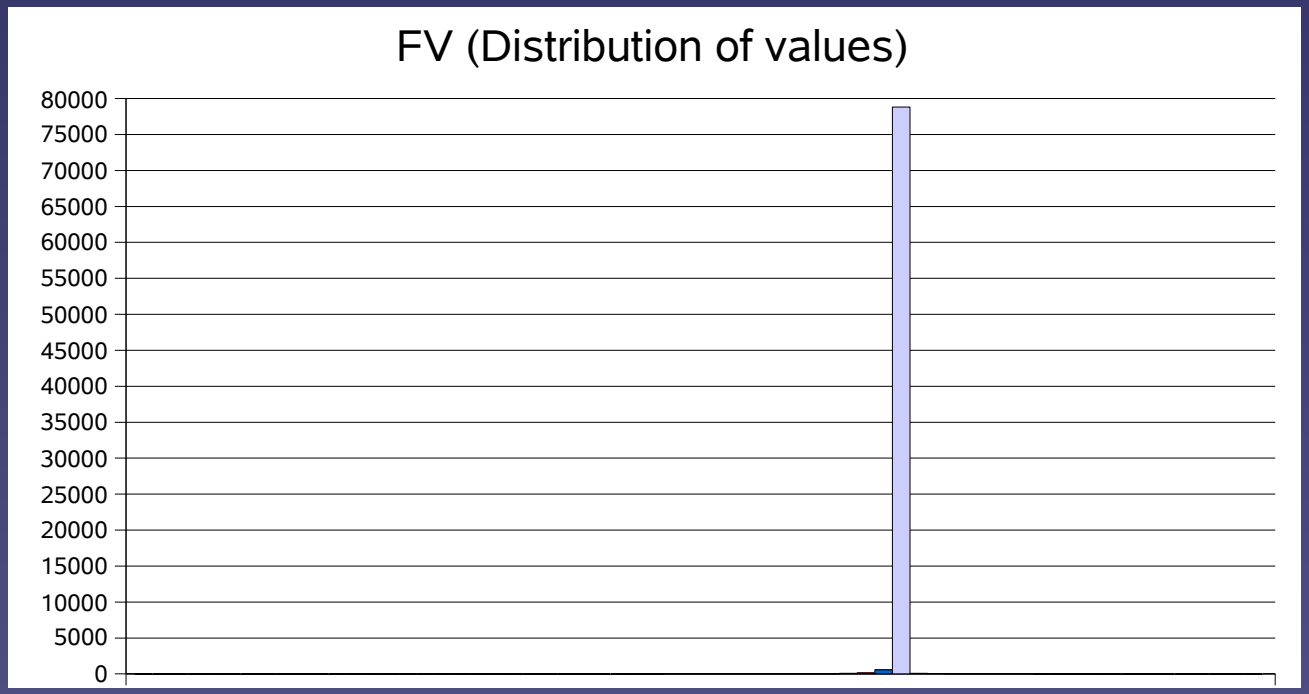


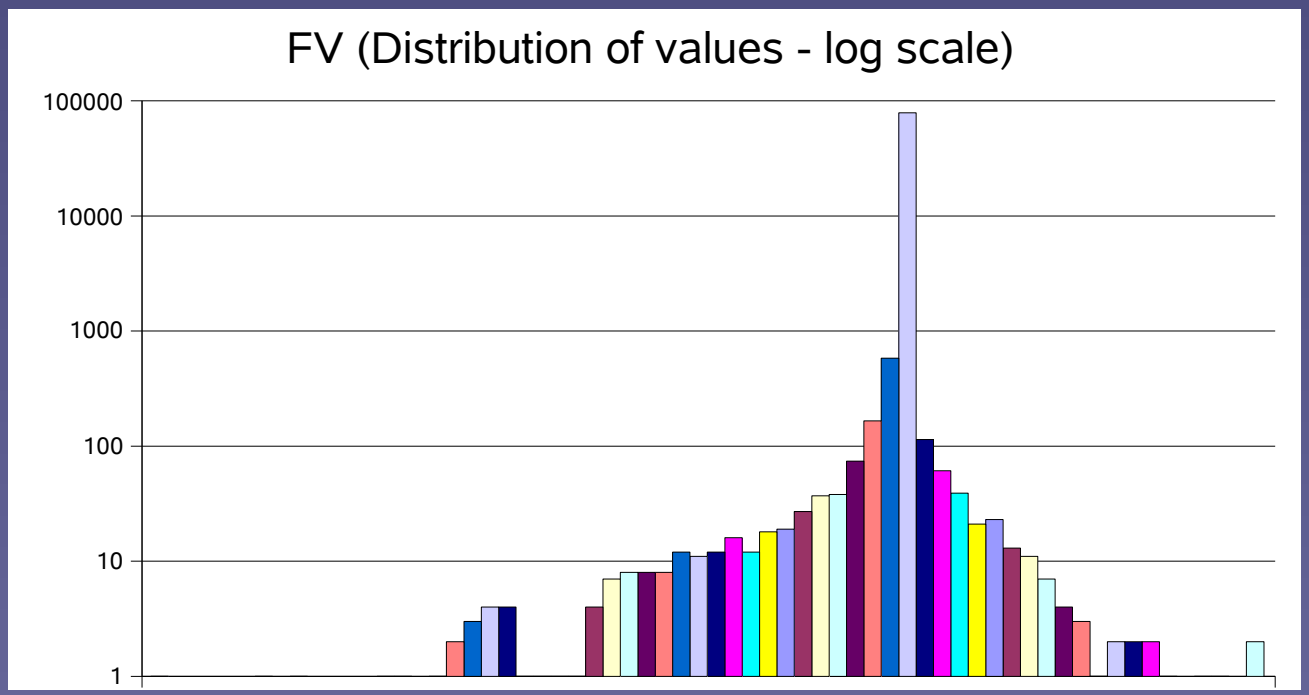
Entropie de quelques champs météo



Prog de 24 hrs FV – niveau 2 : Entropie : 0.636

Distribution des valeurs (en tranches de 64)





Quelques observations

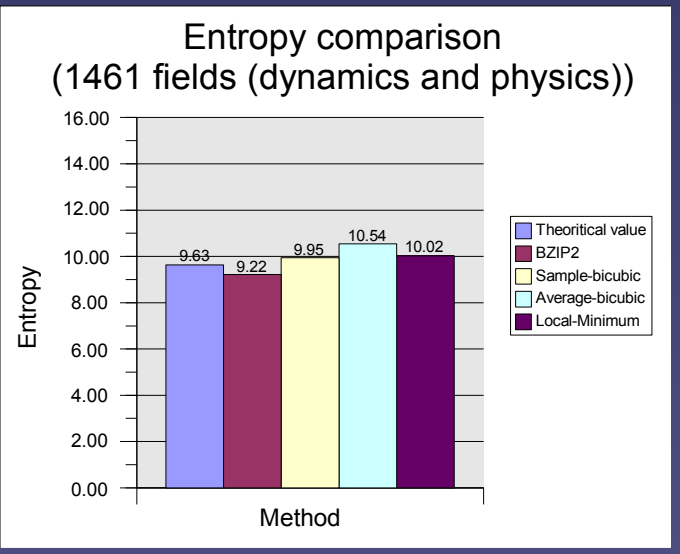
- Champs à grande entropie (potentiel de compression mitigé)
  - HU, HR, UU, VV, WW
- Champs à entropie moyenne (potentiel de compression intéressant)
  - GZ, TT, PN
- Champs à faible entropie (potentiel de compression excellent)
  - Les champs de la physique (FV, FQ)

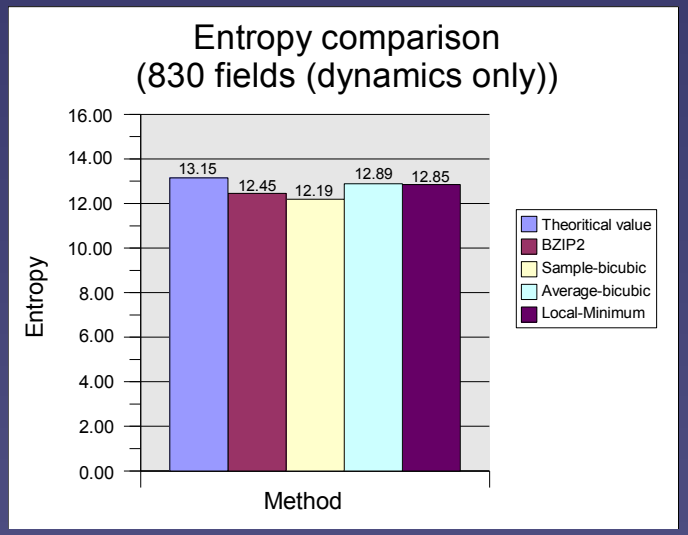
Comment matérialiser l'entropie ?

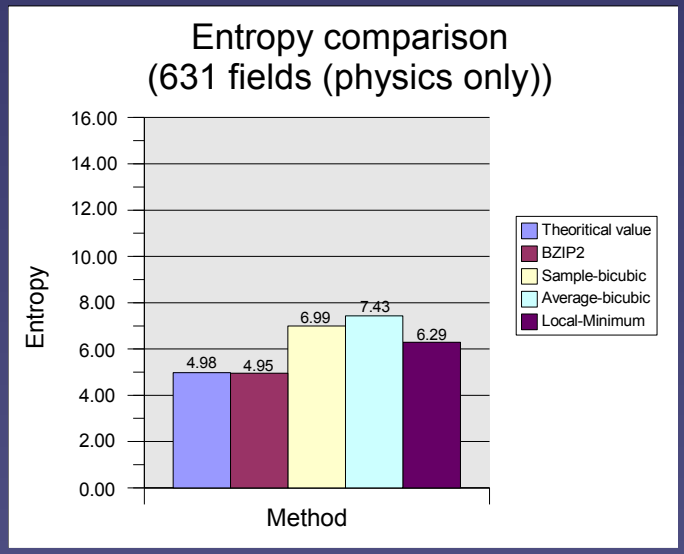
- Avec un encodeur arithmétique
- Espèce de moulin à viande qui reçoit des jetons en entrée et produit une séquence de bits de longueur quasi infinie
- Relativement compliqué a coder
- Les compresseurs de type **bzip2** donnent une performance équivalente

Quelques statistiques

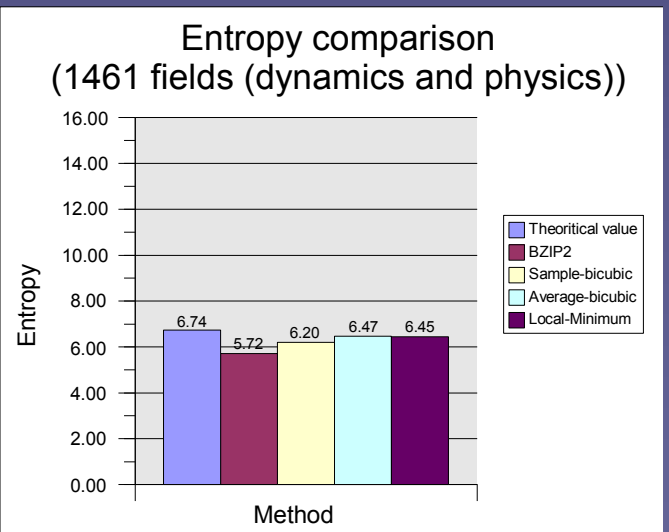
16 bits

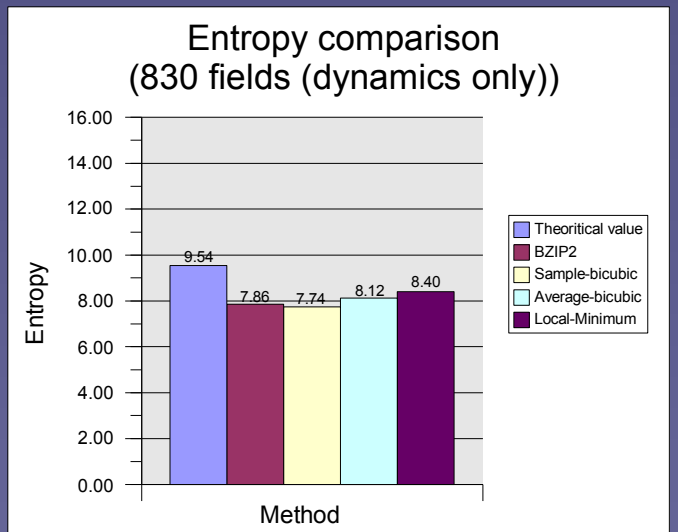


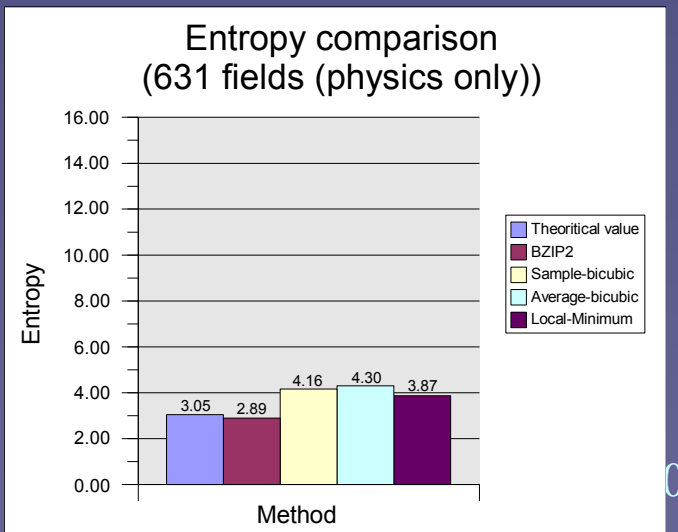




12 bits



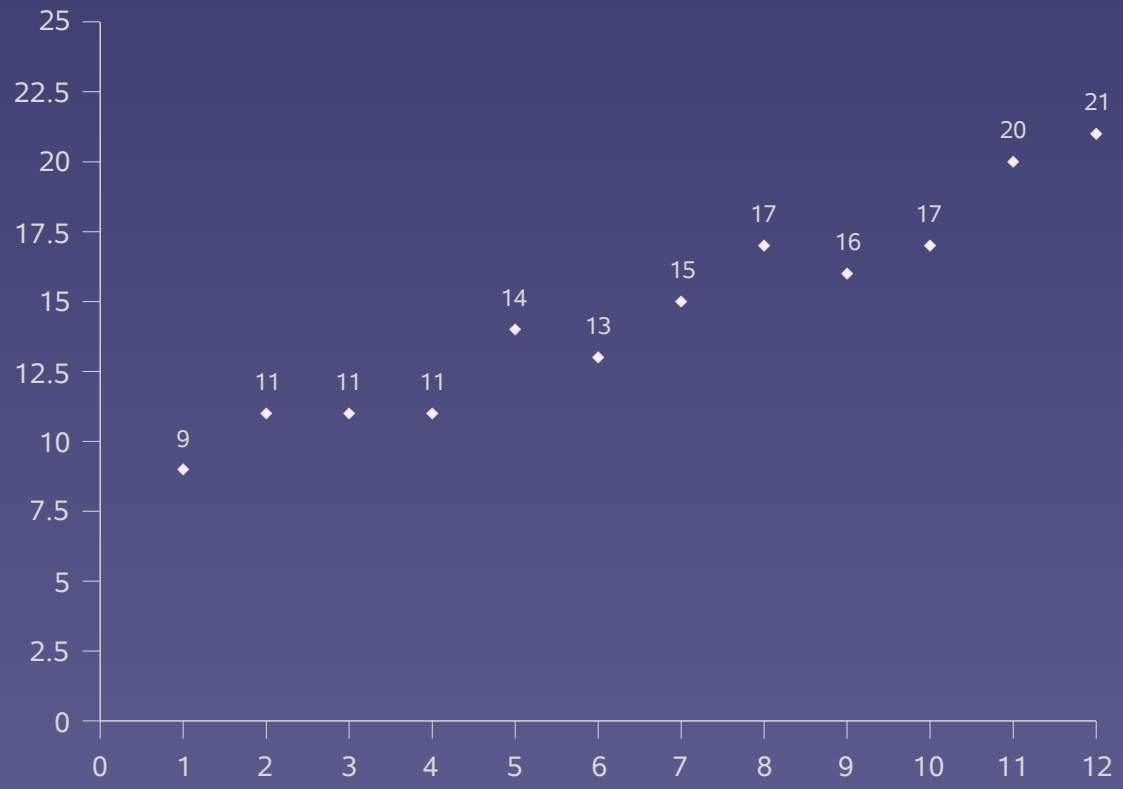




Methode prédicteur-correcteur

- Si on a un modèle pour décrire le comportement des données, on peut d'autant plus augmenter notre potentiel de compression.
- Il ne suffit que d'encoder les erreurs de prédiction (les résidus), idéalement avec un encodeur arithmétique.

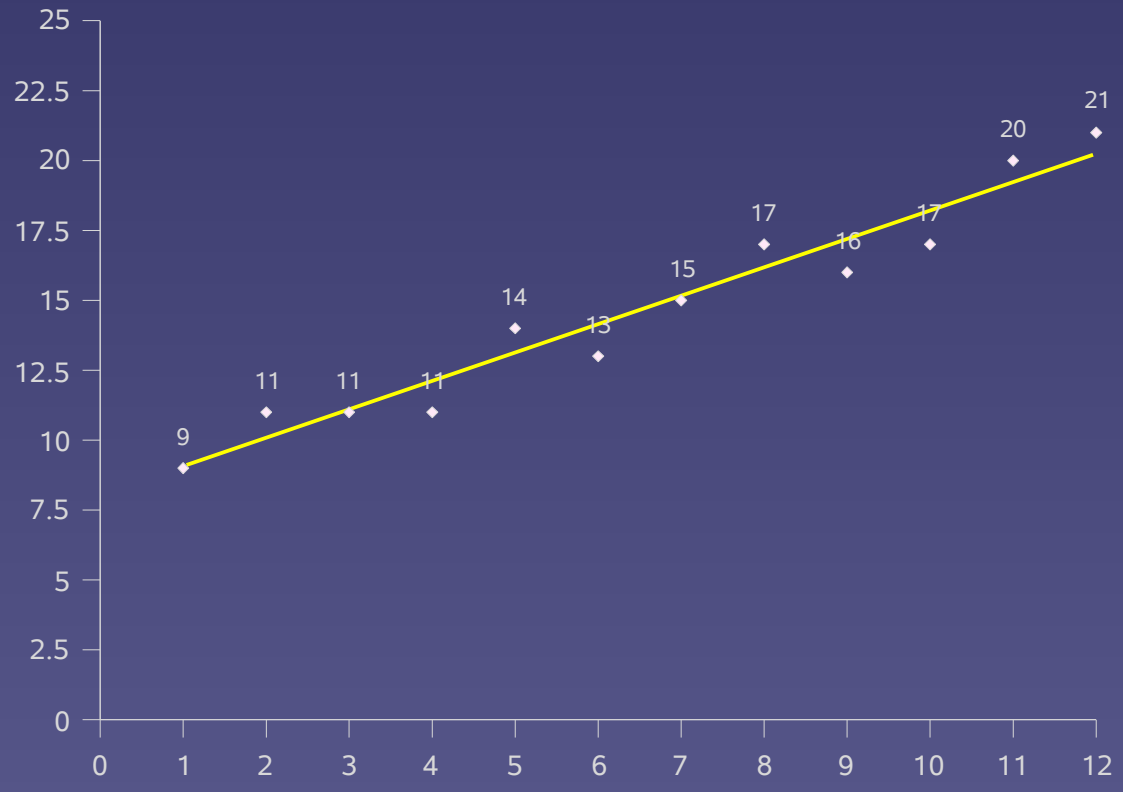
Exemple de compression améliorée avec un modèle prédictif



X	1	2	3	4	5	6	7	8	9	10	11	12
Y	9	11	11	11	14	13	15	17	16	17	20	21

Entropie = 3.19

Exemple de compression améliorée avec un modèle prédictif (2)



Supposons que Y = X + 8

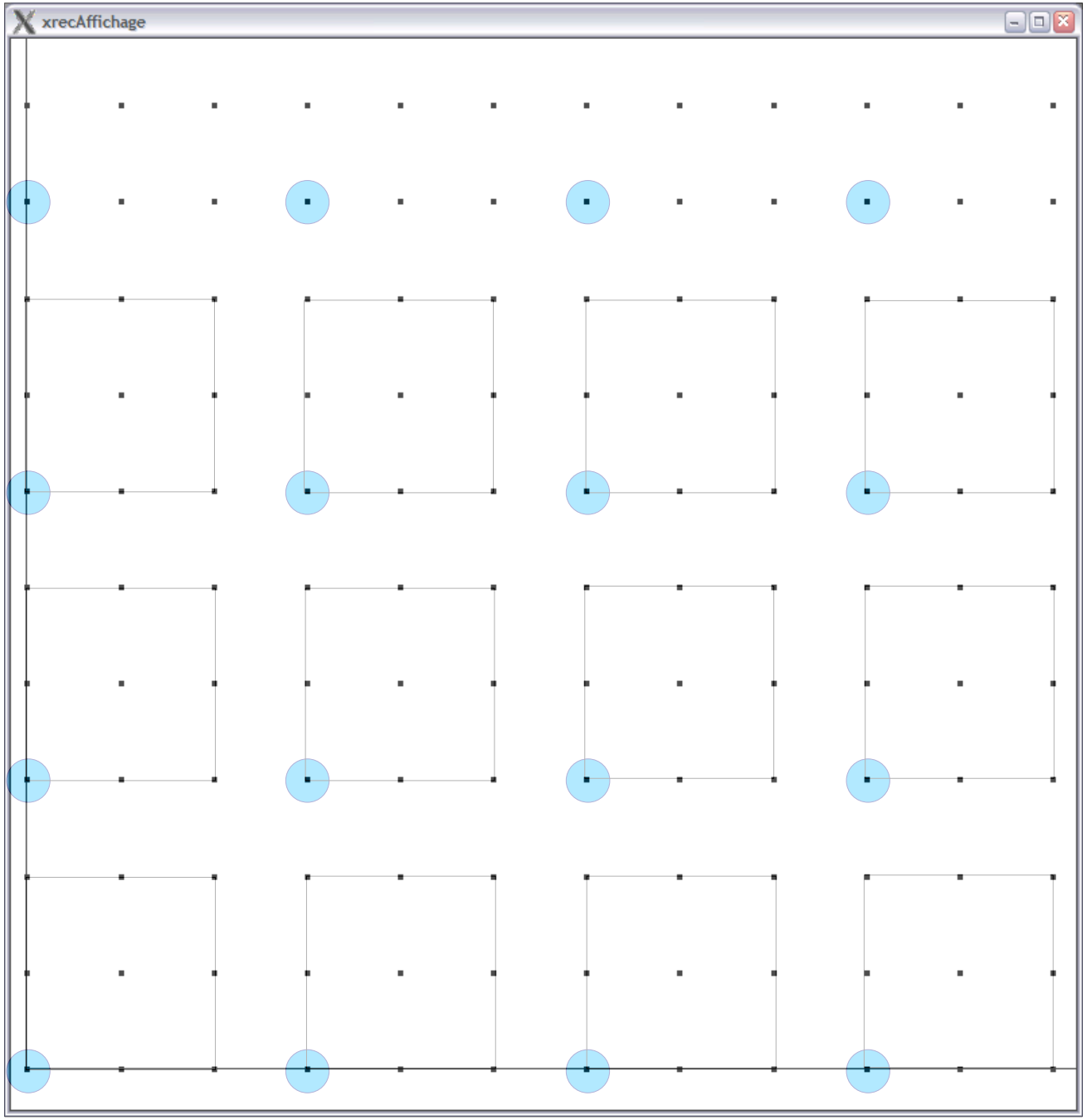
Entropie = 1.58

X	1	2	3	4	5	6	7	8	9	10	11	12
Y	9	11	11	11	14	13	15	17	16	17	20	21
P	9	10	11	12	13	14	15	16	17	18	19	20
E	0	-1	0	1	-1	1	0	-1	1	1	-1	-1

Approche générale

- La base de la partie “prédicteur” sera donnée par des fonctions d'interpolation calculées à partir d'une grille “coarse”, échantillonée à tous les N points de grille.
- On utilise ces points “coarse” pour prédire les valeurs à l'intérieur de la grille “fine”.
- On garde les points de la grille “coarse” tels quels.
- On encode uniquement les erreurs de prédiction sur les points de la grille fine.

Grille coarse et grille fine



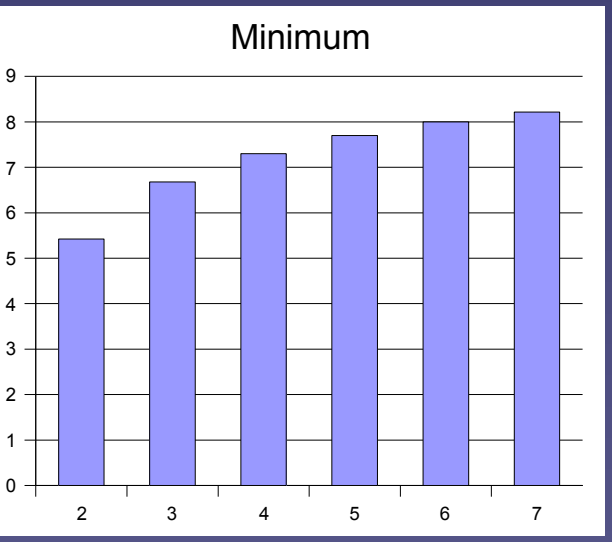
Prédicteurs utilisés

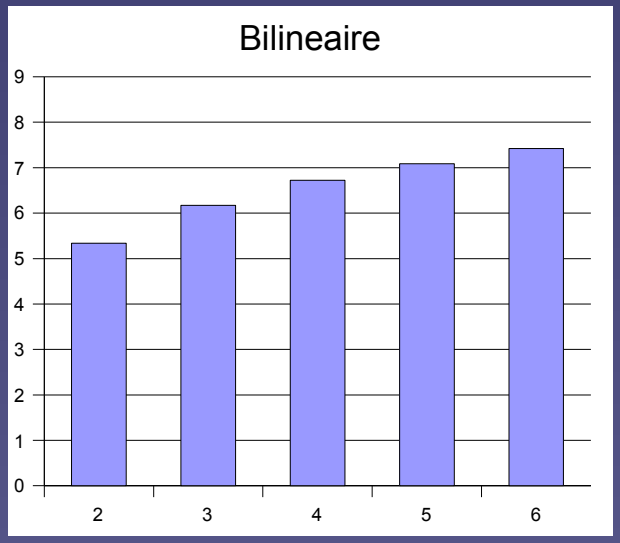
- Degré des prédicteurs
  - Voisin (minimum)
  - Bi-linéaire (4 points)
  - Bi-cubique (16 points)
- Espacement des points
  - 2,3,4,5,6,7
- Utilisation des points
  - “Sample”
  - Moyenne

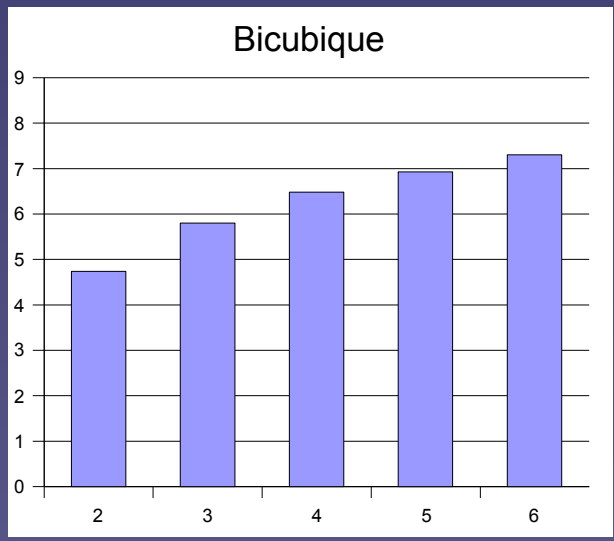
Compromis

- Utiliser N pour la grille coarse le plus grand possible (minimise le nombre de points que l'on garde)
- Minimiser les erreurs de prédiction le plus possible, en sachant que les erreurs augmentent à mesure que N augmente.

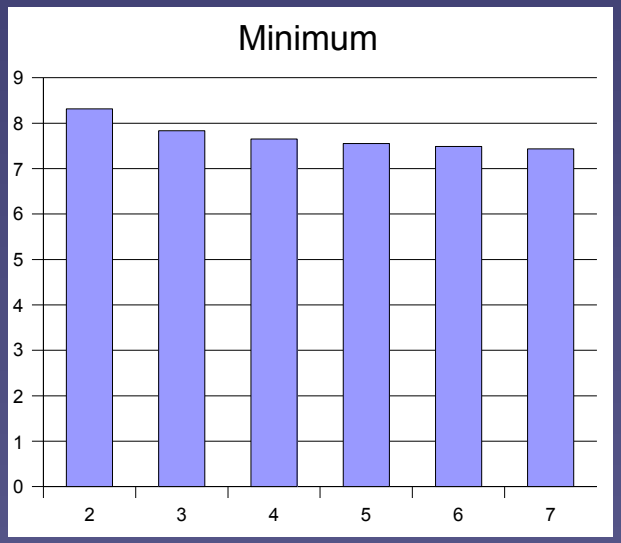
Entropie des erreurs de prédiction selon la méthode et l'espacement entre les points (Moyenne prog REGPRES 24 hrs)

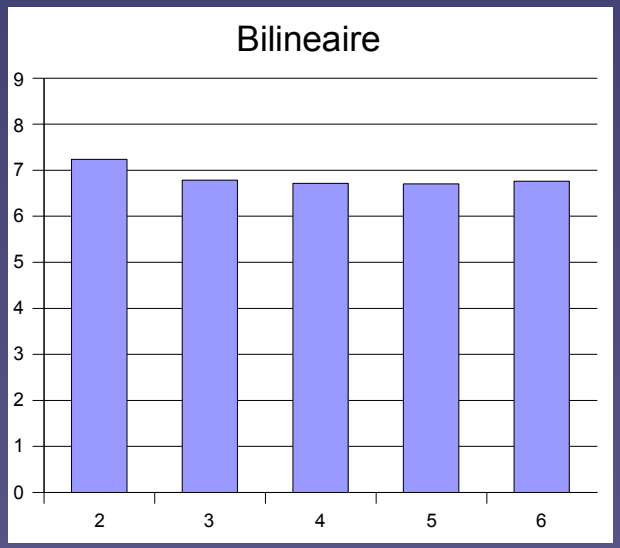


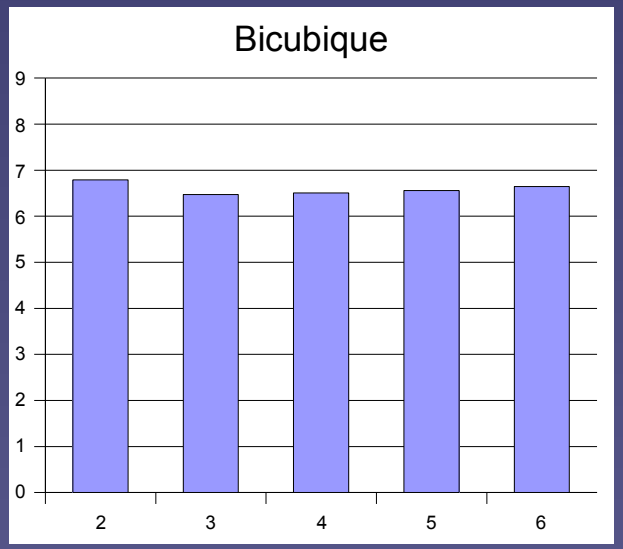




Entropie totale des champs selon la méthode et l'espacement entre les points (Moyenne prog REGPRES 24 hrs)







Encodage des résidus

- Plusieurs schémas (plus ou moins tordus) ont été envisagés
- 2 encodages ont survécu à une 1e passe
  1) Encodage des résidus avec la méthode **bzip2**
  2) Encodage des résidus avec le nombre minimum de bits requis pour les exprimer
- L'encodage de type 2 a été finalement choisi
  - beaucoup plus rapide
  - donne des résultats comparables en espace
  - ne pose aucun problème concernant la propriété intellectuelle

2 méthodes générales ont été retenues

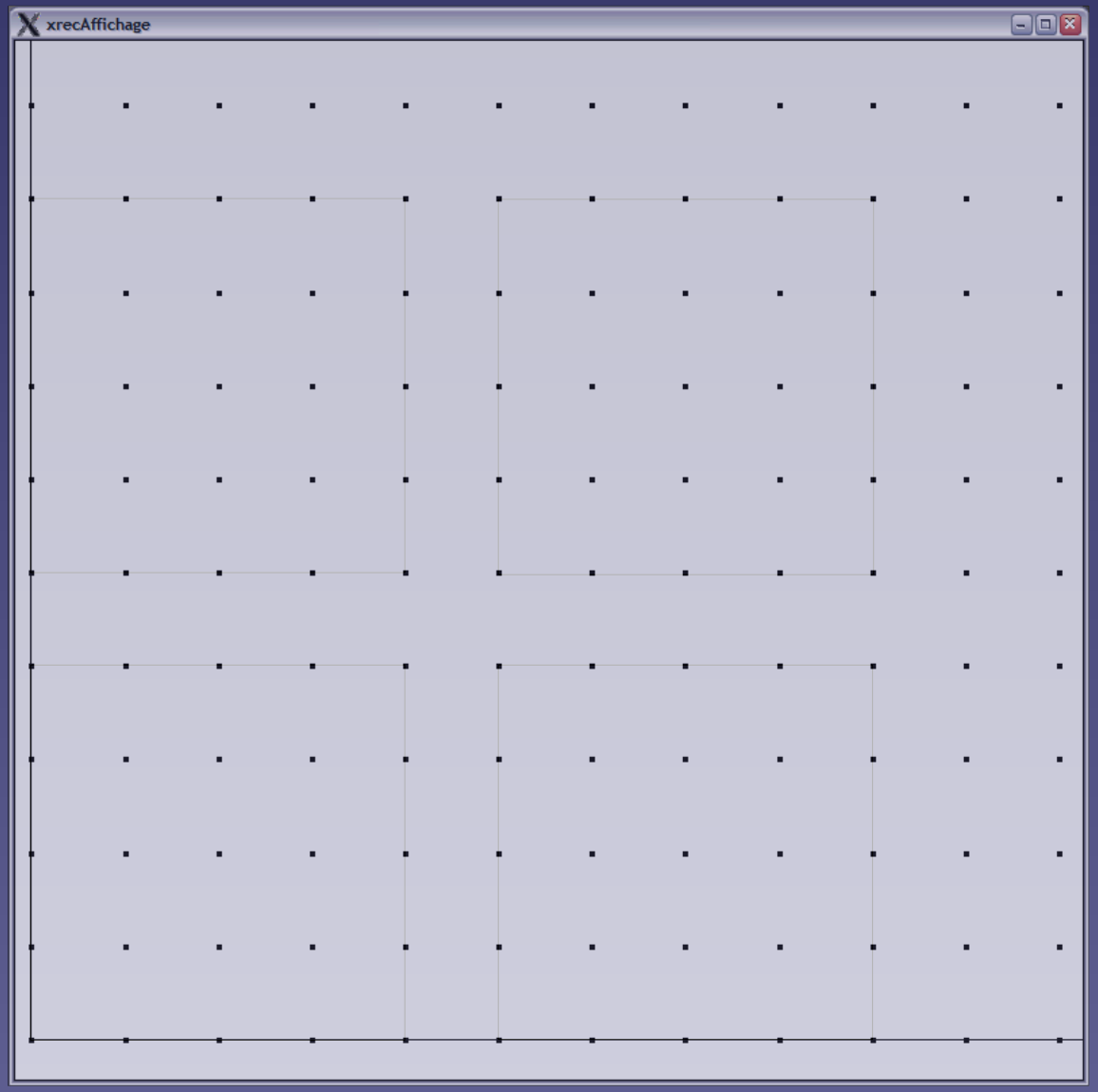
- Compression de type “minimum” avec tuilage 5x5
- Compression de type bicubique avec tuilage 3x3
- **Activées lorsque nbits <= 16**

2 méthodes générales ont été retenues (Addendum 2006)

- Compression de type “minimum” avec tuilage 5x5
- Compression de type Lorenzo avec tuilage 3x3
- **Activées lorsque nbits <= 16**

- **La méthode bicubique n'est plus utilisée**

La méthode “minimum”



La méthode minimum

40373	40415	40417	40340	40254
40515	40537	40498	40389	40240
40665	40659	40551	40659	40551
40812	40727	40565	40331	40565
40936	40726	40474	40166	39804

569	611	613	536	450
711	733	694	585	436
861	855	747	855	747
1008	923	761	527	761
1132	922	670	362	0

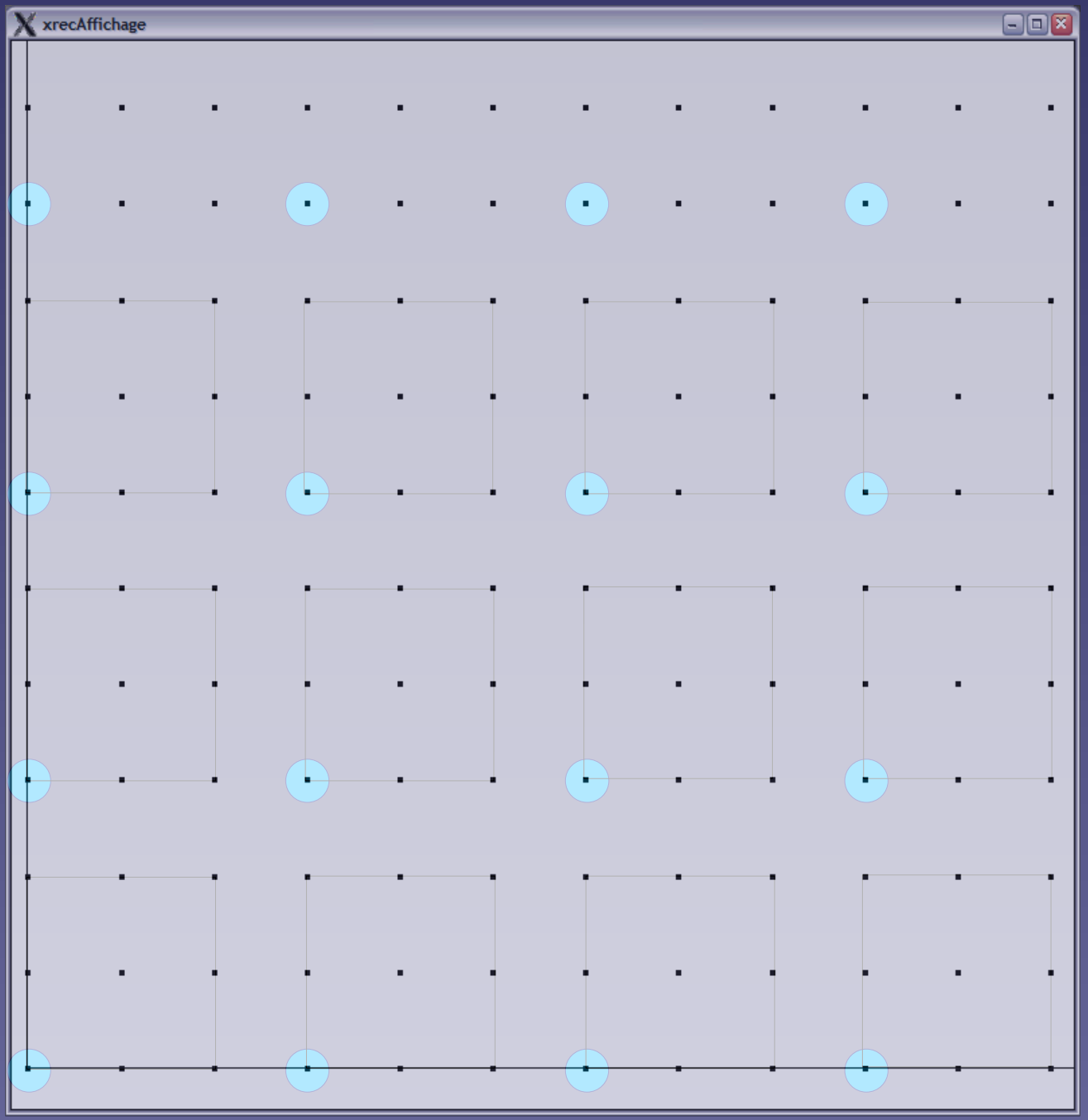
La méthode minimum

569	611	613	536	450
711	733	694	585	436
861	855	747	855	747
1008	923	761	527	761
1132	922	670	362	0

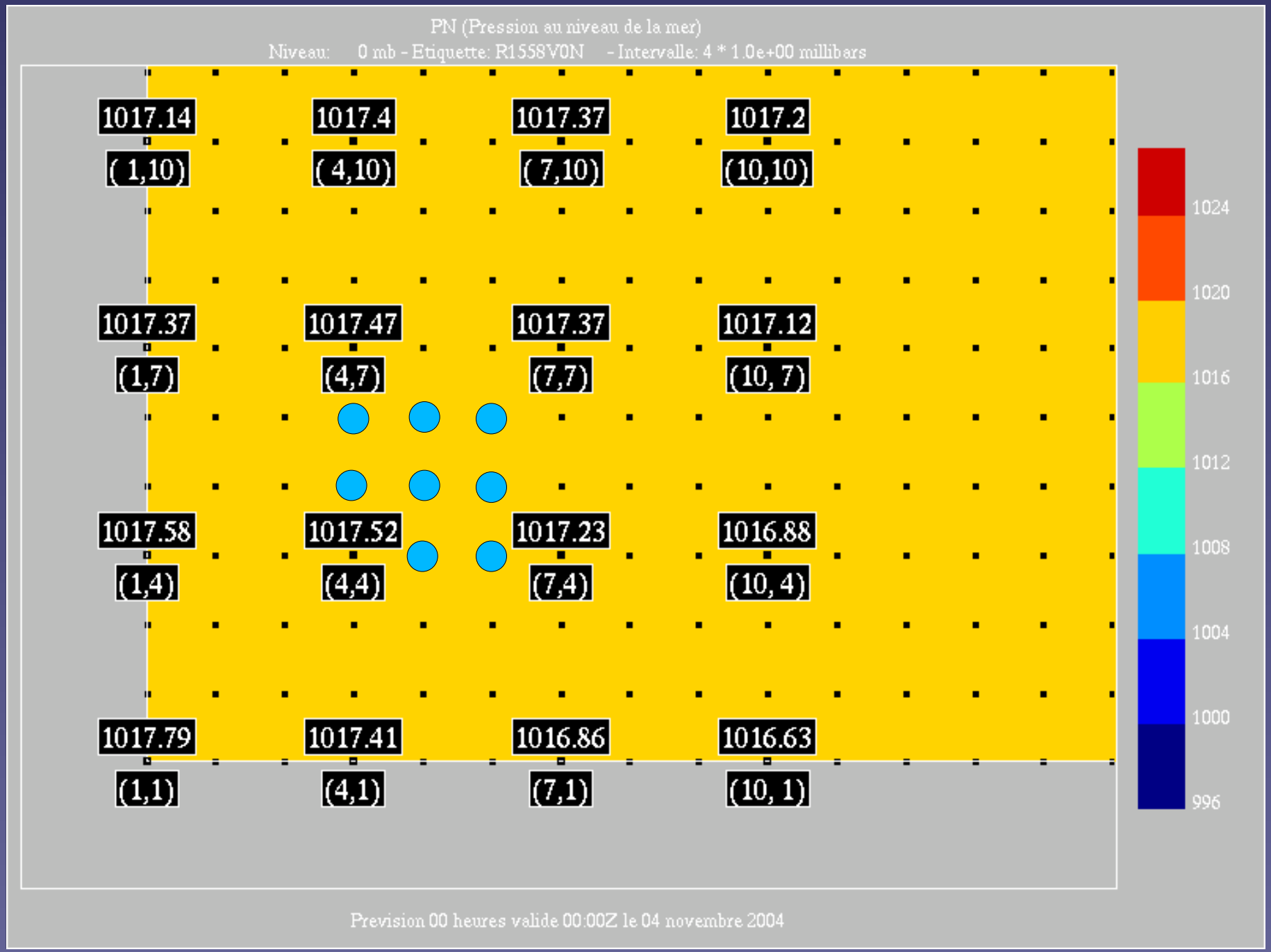
The minimum value is located at point (5,1), the maximum at (1,1).
The tile range is 1132, implying that 11 bits are required to encode this tile ($2^{11} = 2048$).
The bitstream is composed of the following elements :

- nbits_needed (the number of bits needed, from 0 to 15),
- the minimum value of the tile
- the 25 (5x5) encoded differences between the quantized values and the min.

La méthode bi-cubique



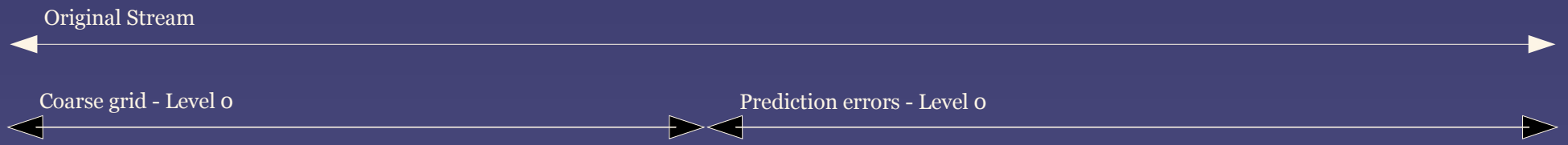
Grille “coarse” et grille fine



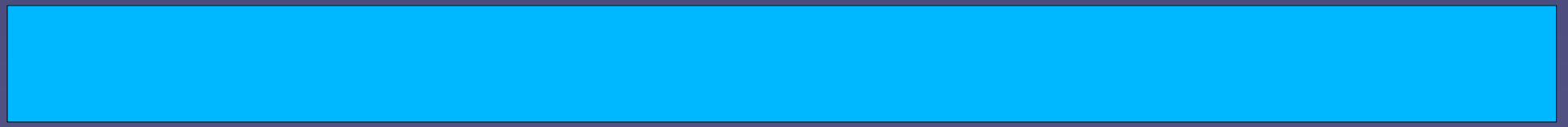
Configuration des points utilisés pour l'interpolation

39604			40148			40093			39743
40077			40286			40092			39581
40515			40389			39794			39075
40936			40166			39046			38564

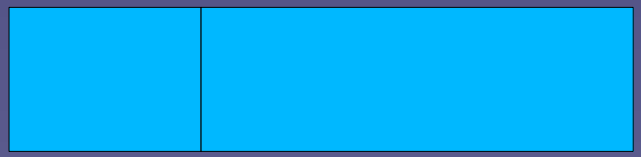
Structure interne des champs compressés avec la méthode bicubique



Champ original

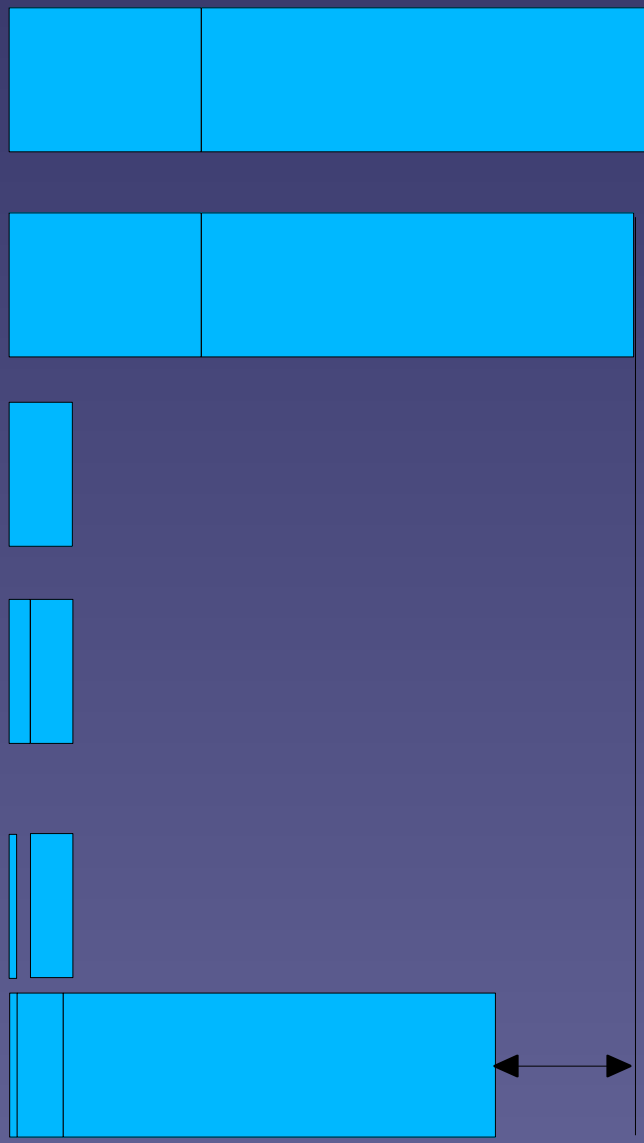


Partie coarse



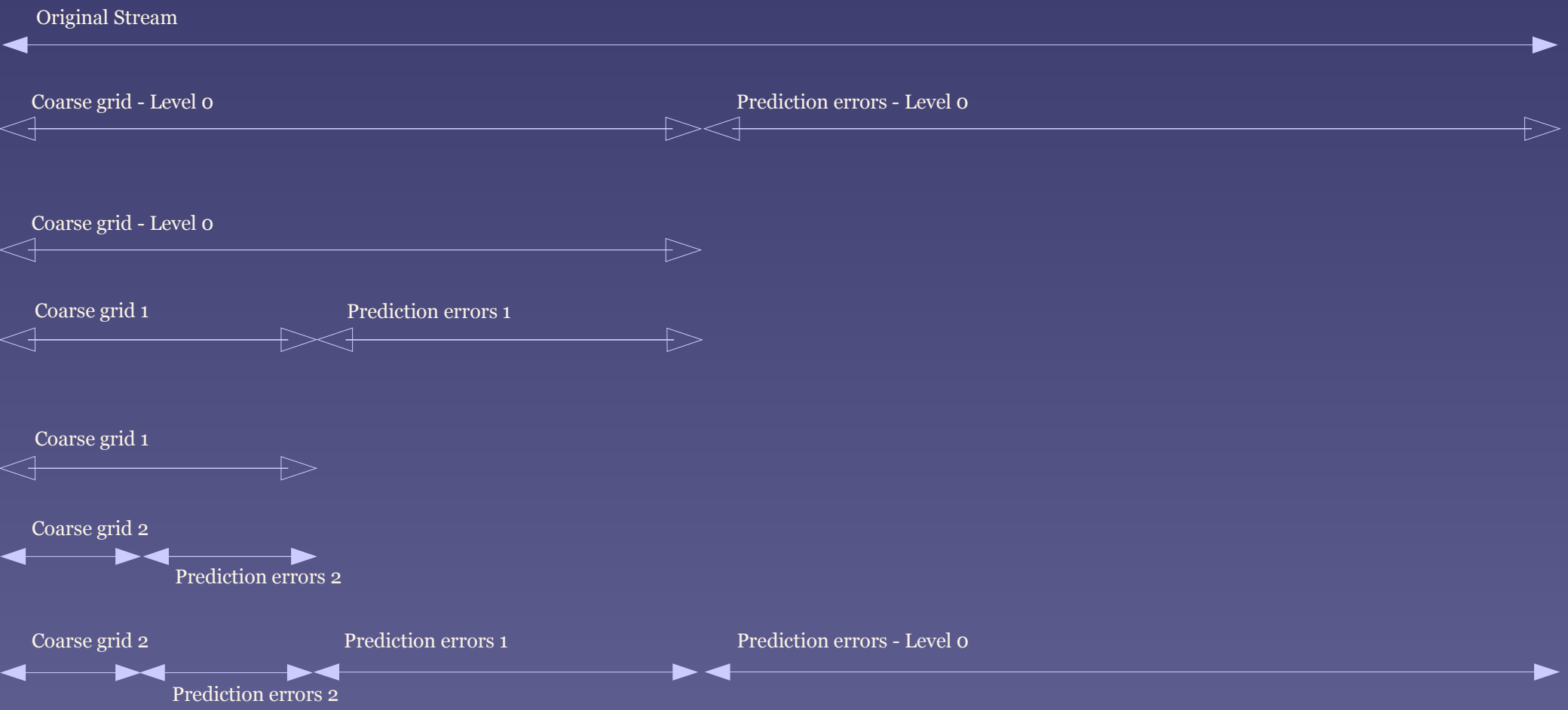
Partie compressée

Compresser un peu plus (suite)



24 % de sauvés !
(si un ratio de 4 est observé pour les erreurs de prédiction

Structure interne des champs compressés avec la méthode bicubique à 3 niveaux



Interpolateur bicubique traditionnel

```
parameter (one = 1.0D0)
parameter (three = 3.0D0)
parameter (six = 6.0D0)
parameter (sixth = one/six)
parameter (third = one/three)
real*8 cubic, dx,dy,z1,z2,z3,z4


cubic(z1,z2,z3,z4,dx)=((((z4-z1)*sixth +
    0.5*(z2-z3))*dx  +
    0.5*(z1+z3)-z2)*dx +
    z3-sixth*z4-0.5*z2-third*z1)*dx+z2
```

dx = 0.3333 ou 0.6666

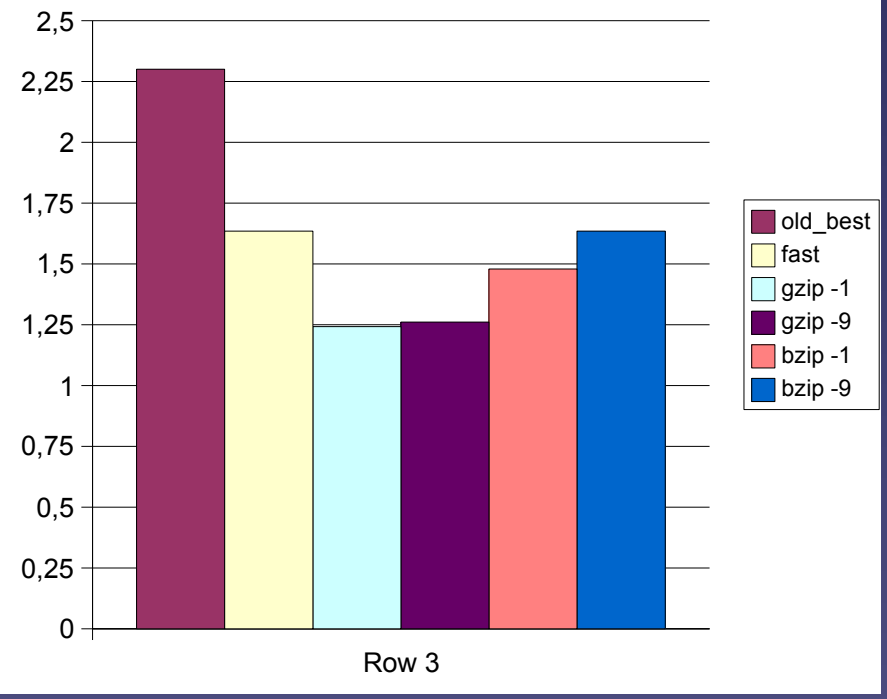
Interpolateur cubique optimisé pour une sous-grille 3x3

```
parameter (fac1 = 108.0D0) ! 12 * 3 * 3
parameter (fac2 = 1944.0D0) ! 72 * 3 * 3 * 3
parameter (unsurfac2 = 1.0D0/fac2) ! 1 / (72 * 3 * 3 * 3)


icubic(z1,z2,z3,z4,dx)= int (0.5001 +
  (z2+(dx*(6*(dx*(2*(dx*((z4-z1)+3*(z2-z3)))+
  18*((z1+z3)-2*z2)))+
  fac1*(6*z3-z4-3*z2-2*z1))) * unsurfac2)


dx = 1 ou 2
```

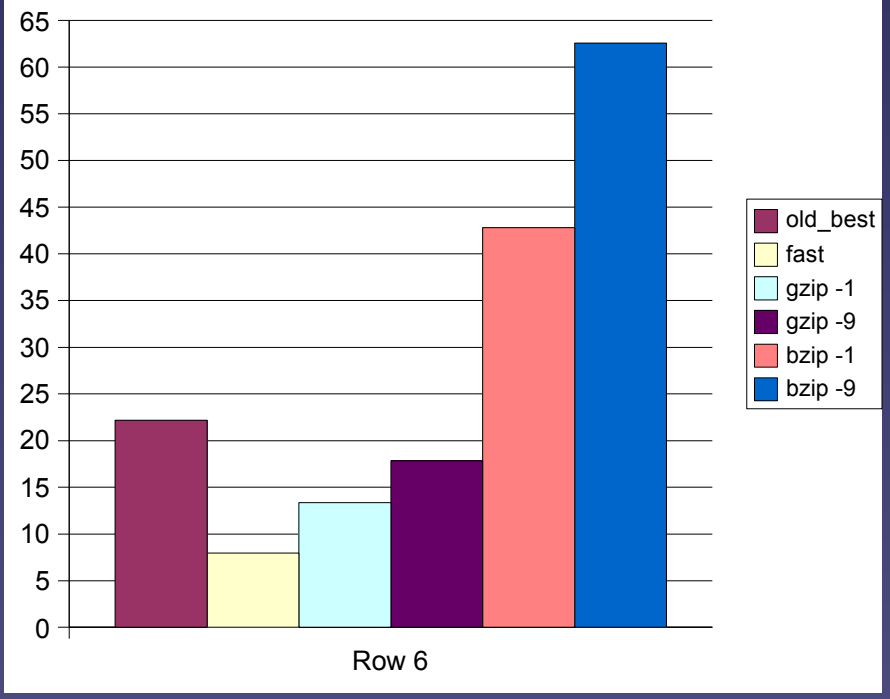
Résultats

- Comparaison d'algorithmes
  - fstcompress -best (meilleur de minimum et bicubique)
  - fstcompress -fast (minimum seulement)
  - gzip -1 (--fast)
  - gzip -9 (--best)
  - bzip2 -1 (--fast)
  - bzip2 -9 (--best)

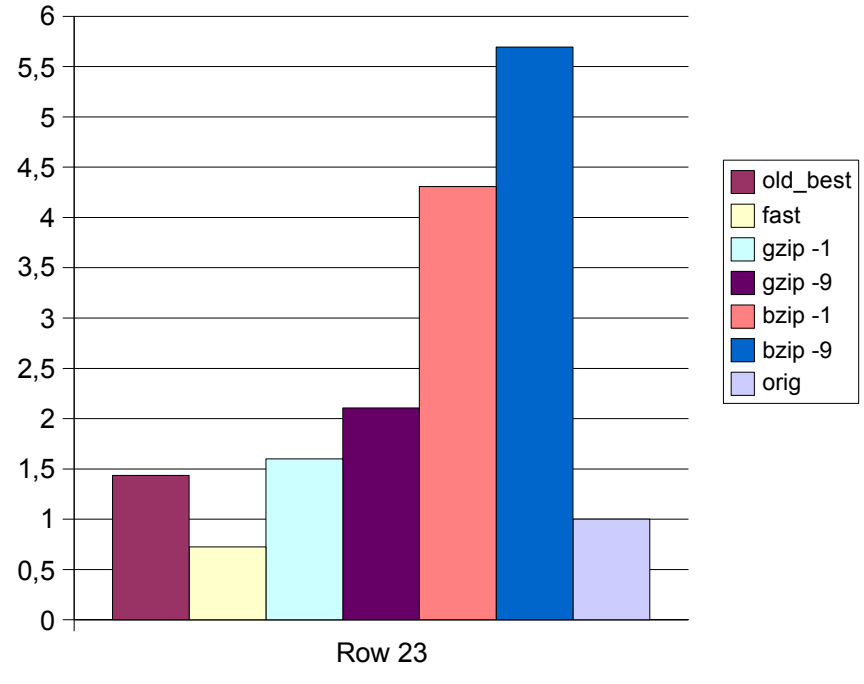
Ratio de compression - regpres00



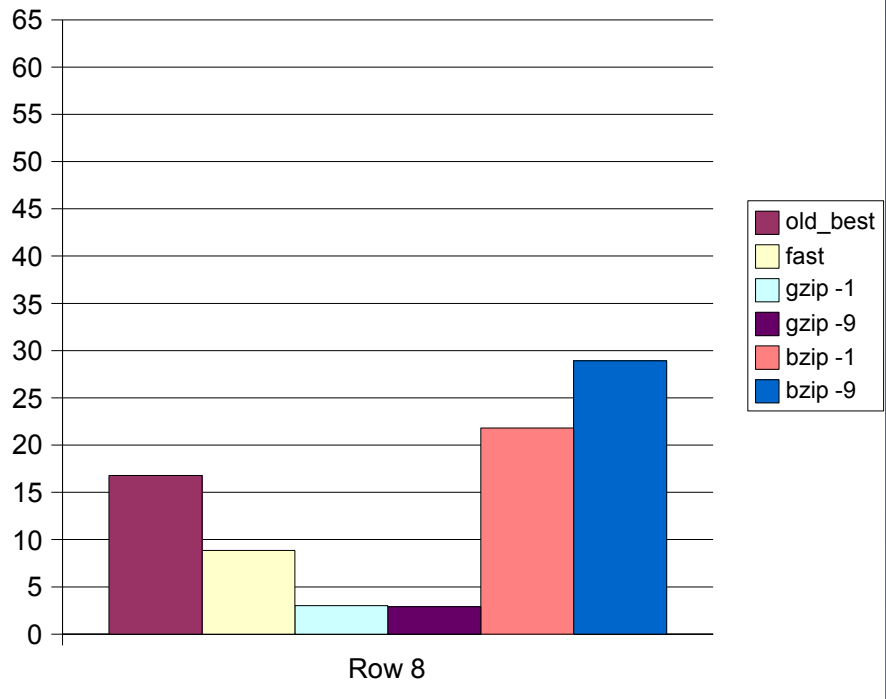
Temps de compression - regpres00



Coût normalisé temps*espace



Temps de decompression - regpres00



Observations (1)

- L'algorithme de compression bicubique donne le meilleur niveau de compression (entre 8 et 30 % fois mieux que bzip2 -best
- Le 2e meilleur est bzip2 –best, mais cet algorithme prend 3 fois plus de temps que la version maison
- Gzip est l'algorithme le moins performant

Observations (2)

- L'algorithme le plus rapide est la version maison avec l'option “fast”
  - Comparable en vitesse à “gzip -1” mais comparable en niveau de compression à “bzip2 –9”
  - 7-10 fois plus rapide à la compression que “bzip2 –9”
  - 2-3 fois plus rapide à la décompression que “bzip2”

Observations (3)

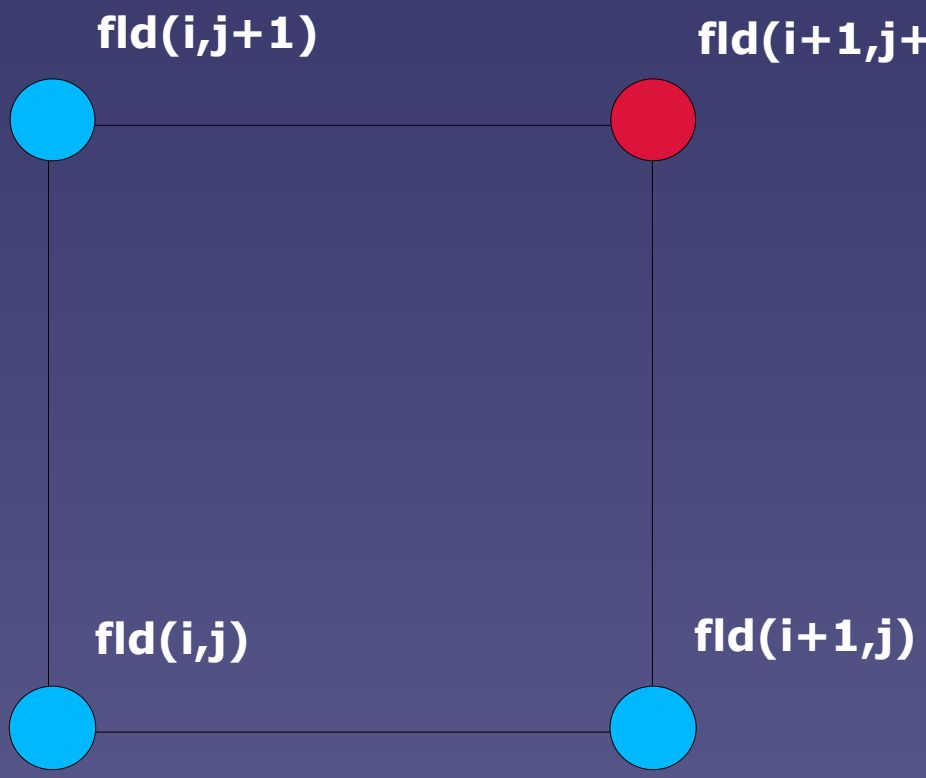
Souvenirs de conférence... revisited

- Vizualisation 2003, Seattle...
- ... l'implantation d'un algorithme prédicteur-correcteur a démontré comment une simulation 4D de 43 Megs pouvait être compressée sans perte à 1.2 Megs (2.5 % de l'original).
- Le fichier regeta (1600 Mbytes à 32 bits) ...
  - Diminue 600 Mbytes à 12 bits...
  - diminue à 398 Mbytes à 8 bits
  - diminue à 104 Mbytes une fois compressé...
  - 6.5 % de l'original

Souvenirs de conférence... revisited 2006

- Une illumination du jeudi matin
- Pourquoi ne pas essayer le potentiel de compression du prédicteur de Lorenzo ?
- On n'a pas grand chose à perdre

Le prédicteur de Lorenzo (en 2-D)



$$fld(i+1,j+1) = fld(i,j+1) + fld(i+1,j) - fld(i,j)$$

Le prédicteur de Lorenzo

Valeurs originales

40373	40415	40417	40340	40254
40515	40537	40498	40389	40240
40665	40659	40551	40659	40551
40812	40727	40565	40331	40565
40936	40726	40474	40166	39804

Valeurs prédites

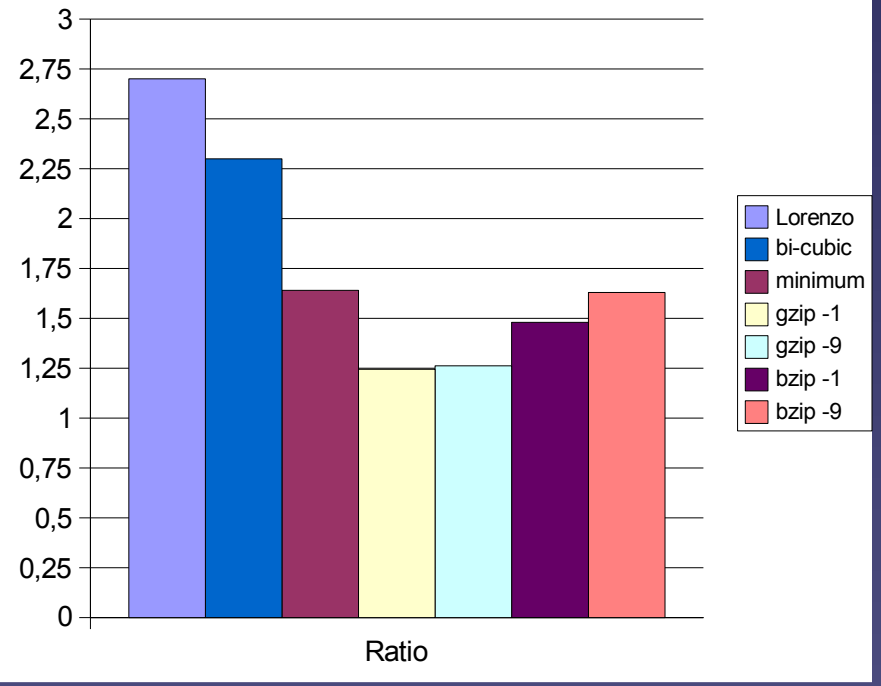
40373	40395	40376	40308	40191
40515	40509	40429	40606	40281
40665	40580	40497	40317	40893
40812	40602	40475	40257	39969
40936	40726	40474	40166	39804

Erreurs de prédiction

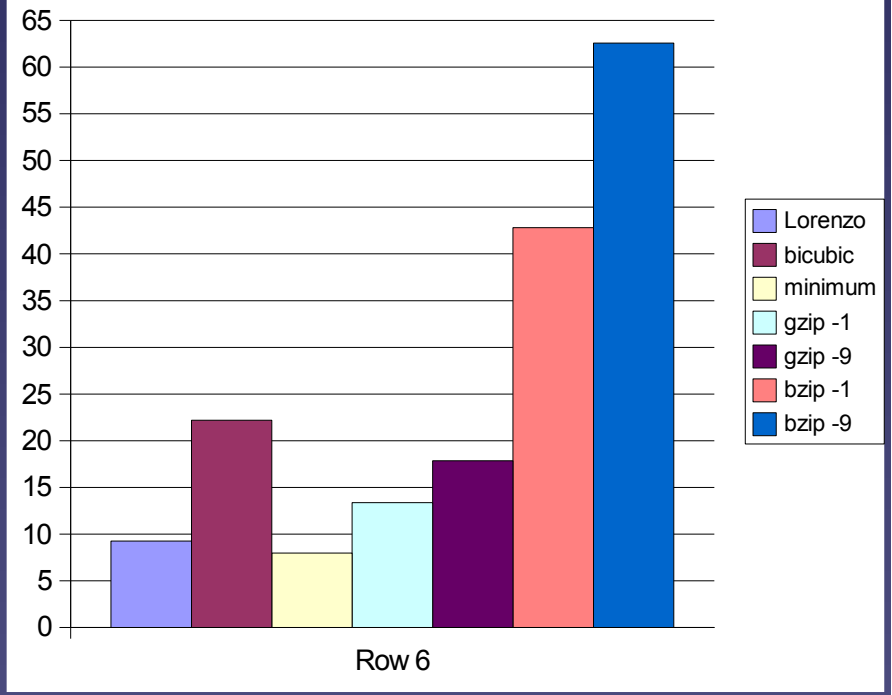
0	-20	-41	-32	-63
0	-28	-69	217	41
0	-79	-54	-342	342
0	-125	-90	-74	-596
0	0	0	0	0

$f(2,2) = f(1,2) + f(2,1) - f(1,1) = 40812 + 40726 - 40936 = 40602$

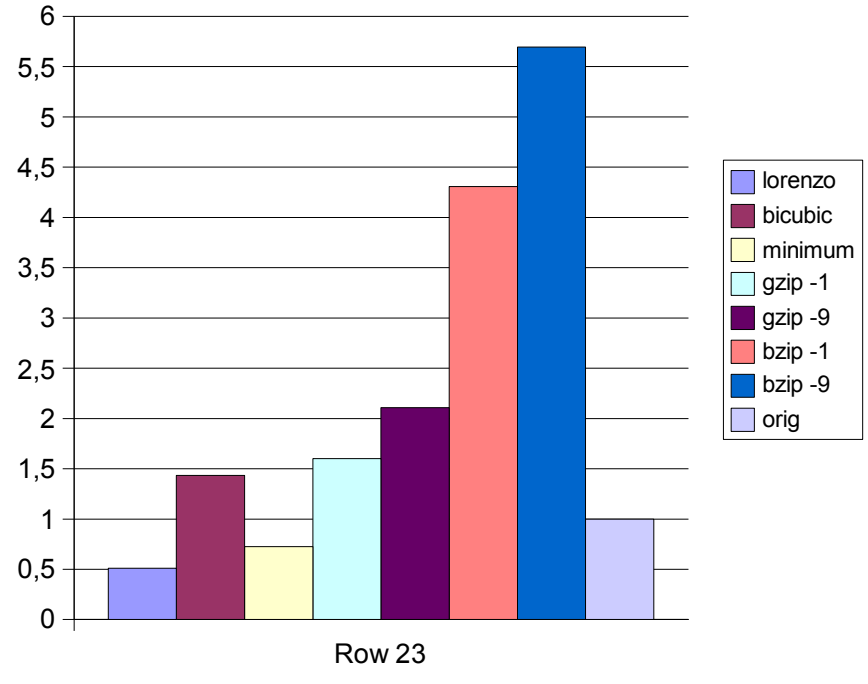
Ratio de compression - regpres00



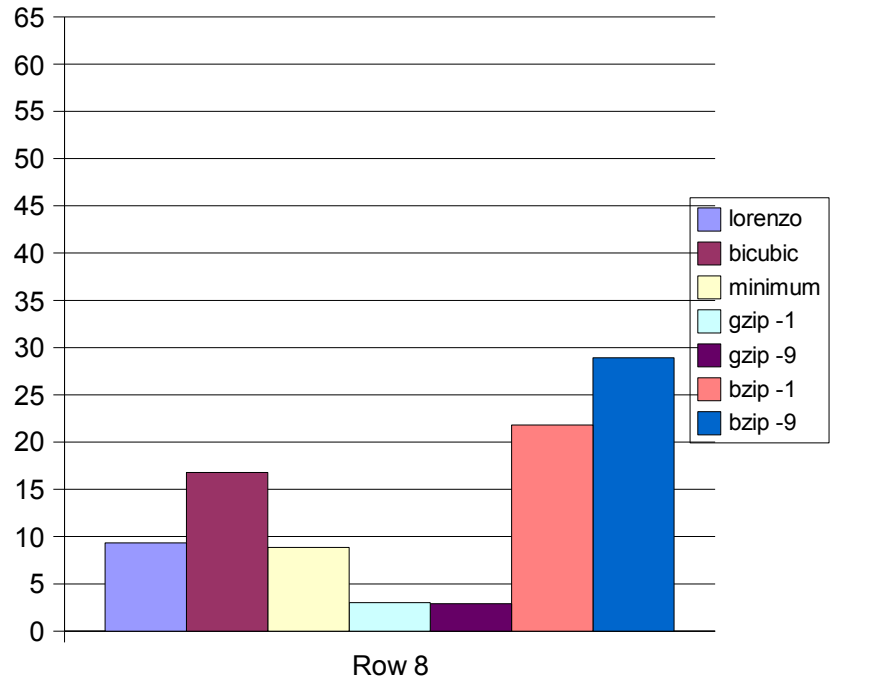
Temps de compression - regpres00



Coût normalisé temps*espace



Temps de decompression - regpres00



Implantation du prédicteur de Lorenzo

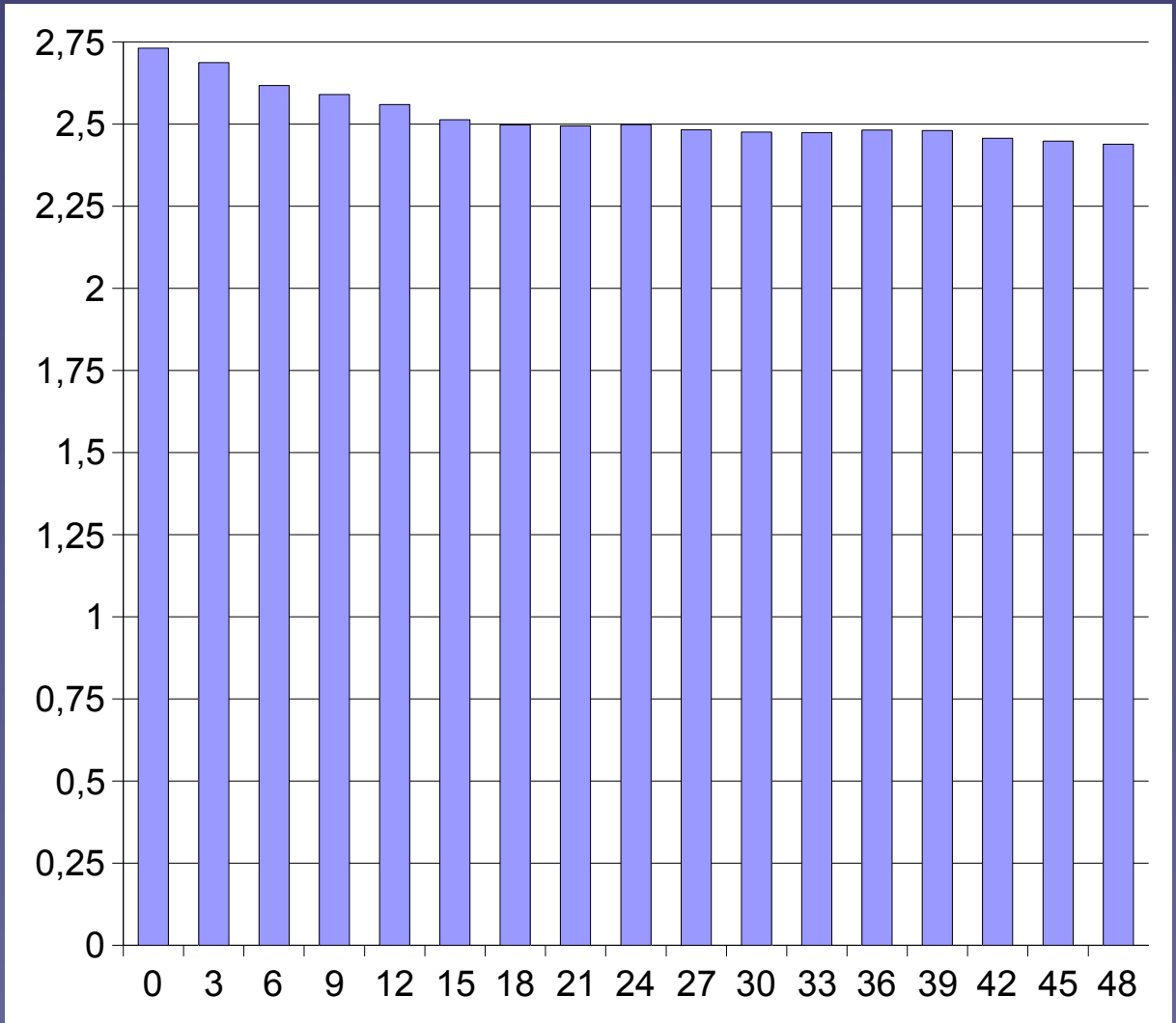
- Faible overhead (0,33%)
  - Ni + nj – 1 points (valeurs originales)
    - 1e rangée de points (1..ni)
    - 1e colonne de points (2..nj)
- On calcule les erreurs du prédicteur de Lorenzo pour chaque point de grille
- Les erreurs sont également distribuées de part et d'autre de zéro. Pas besoin de conserver de valeurs de référence comme pour la methode minimum.
- On groupe les tuiles en carrés de 3x3.

Pourquoi le prédicteur de Lorenzo est-il si bon ?

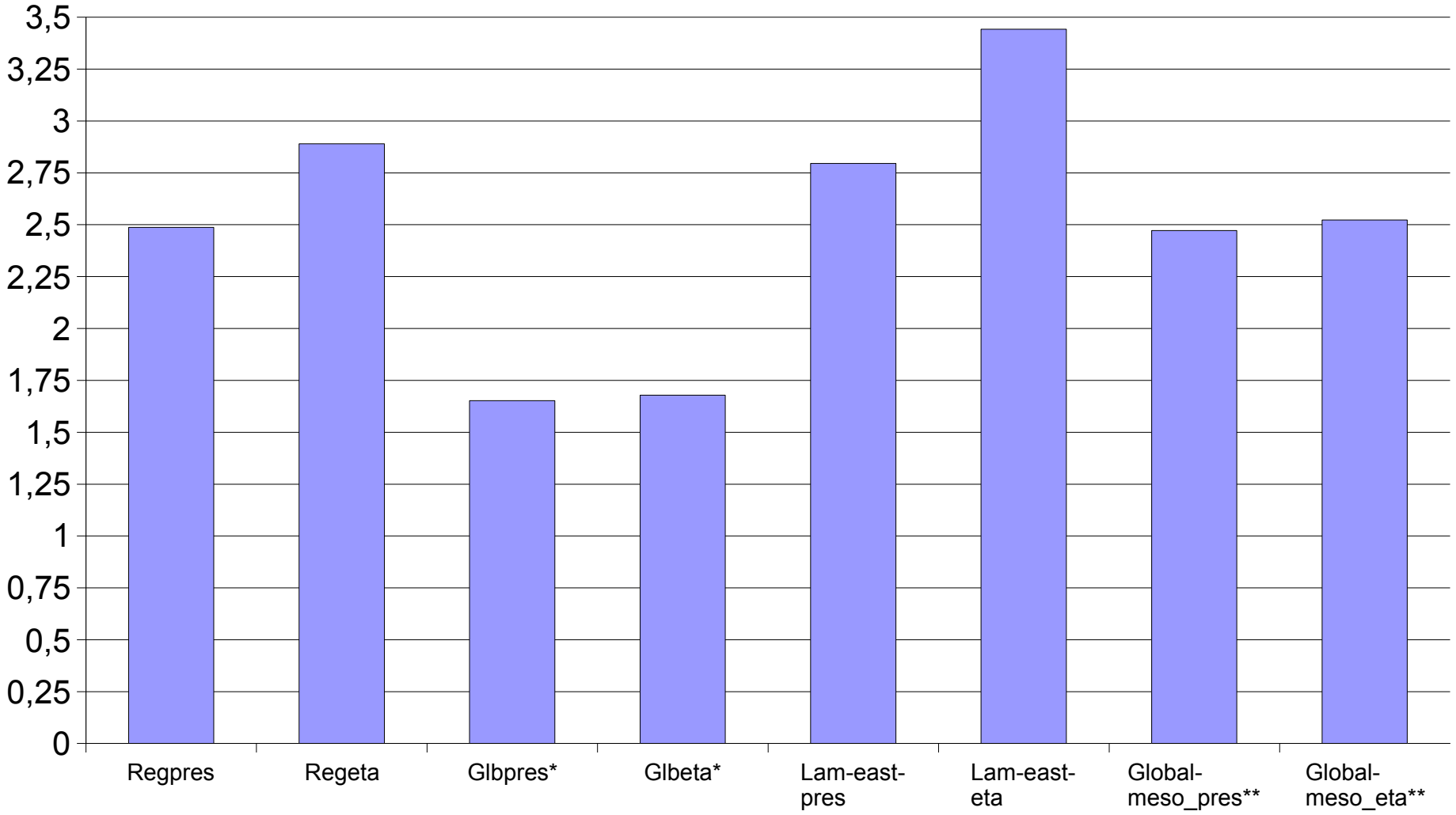
- Les erreurs de prédiction sont comparables à la méthode bi-cubique (l'effet de proximité joue en faveur de Lorenzo).
- L'avantage de Lorenzo pour le stockage est son très faible overhead
- L'avantage de Lorenzo pour le temps de traitement est le très faible coût de calcul de prédiction (2 opérations entières comparativement à 125 opérations point flottant pour la bi-cubique).

Évolution du ratio de compression avec l'heure de prévision

(sorties opérationnelles du GEM 15 km - REGPRES)

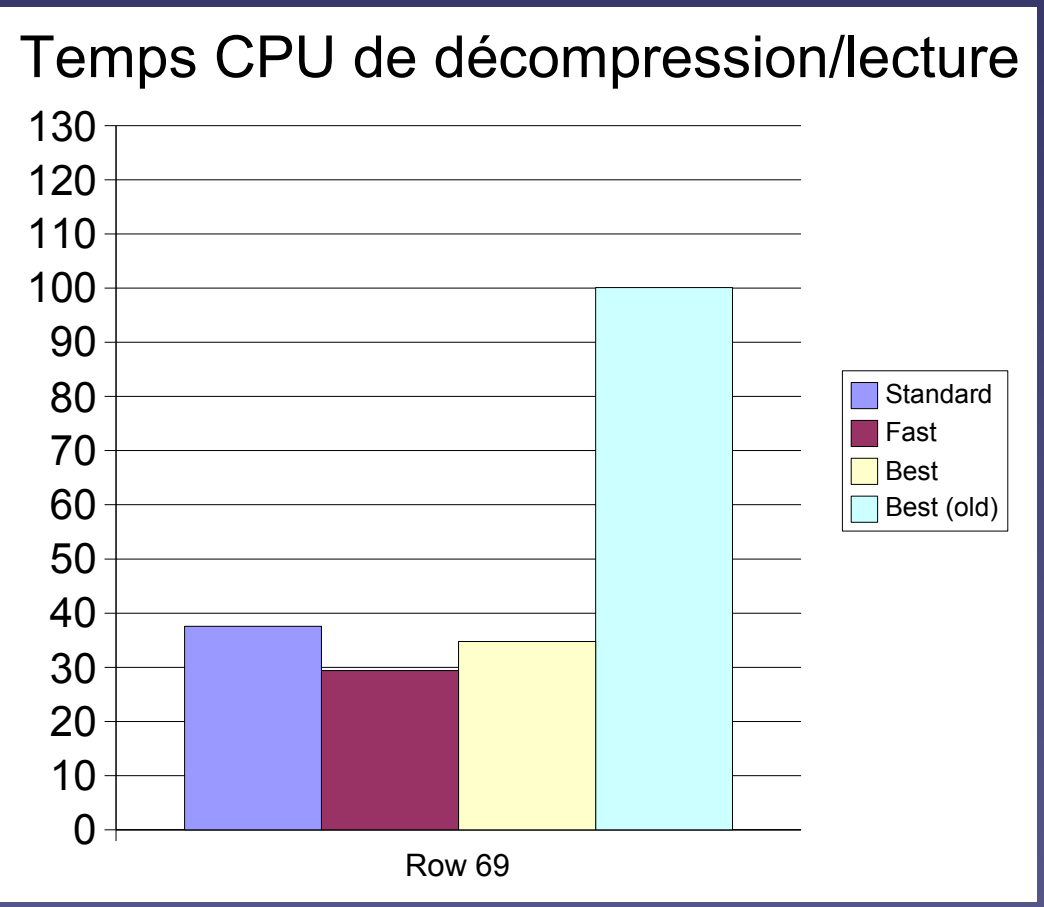


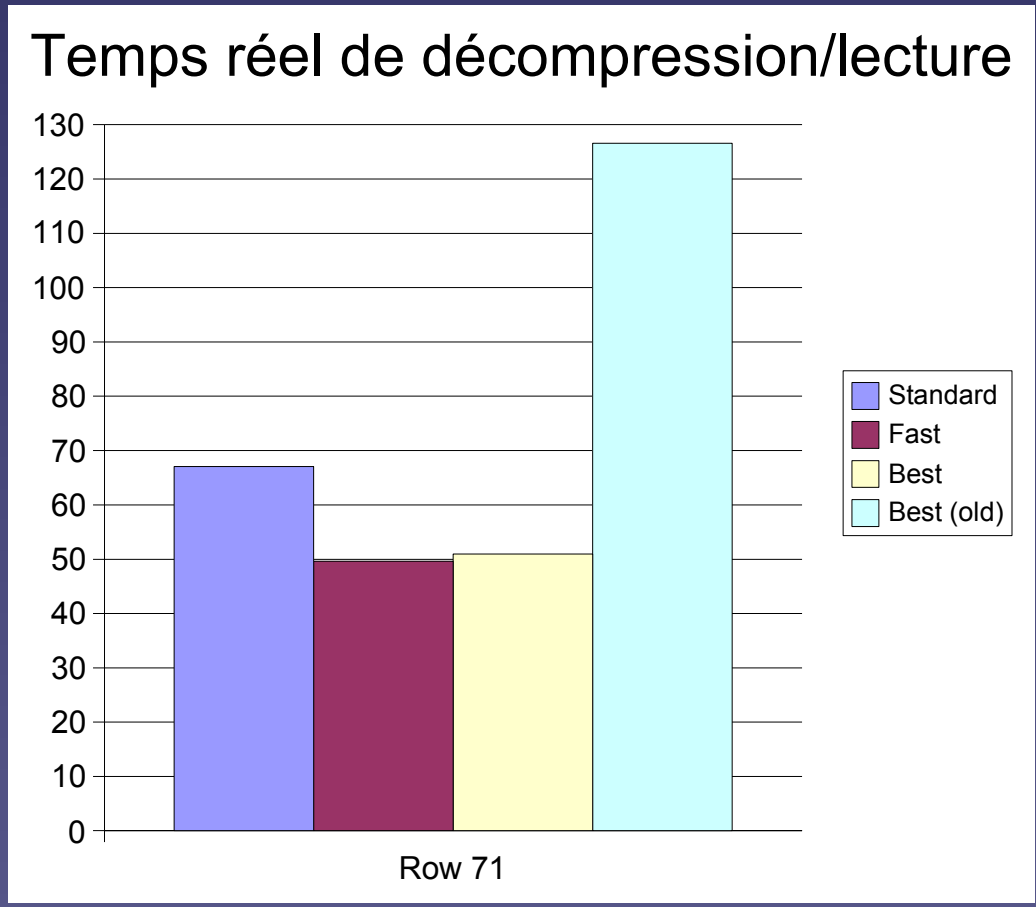
Ratios de compression pour divers fichiers opérationnels



*Les sorties du modèle global ont été ramenées à R16 avant compression
**Les sorties du meso-global ont ramenées à R12 avant compression

Overhead de lecture de champs compressés

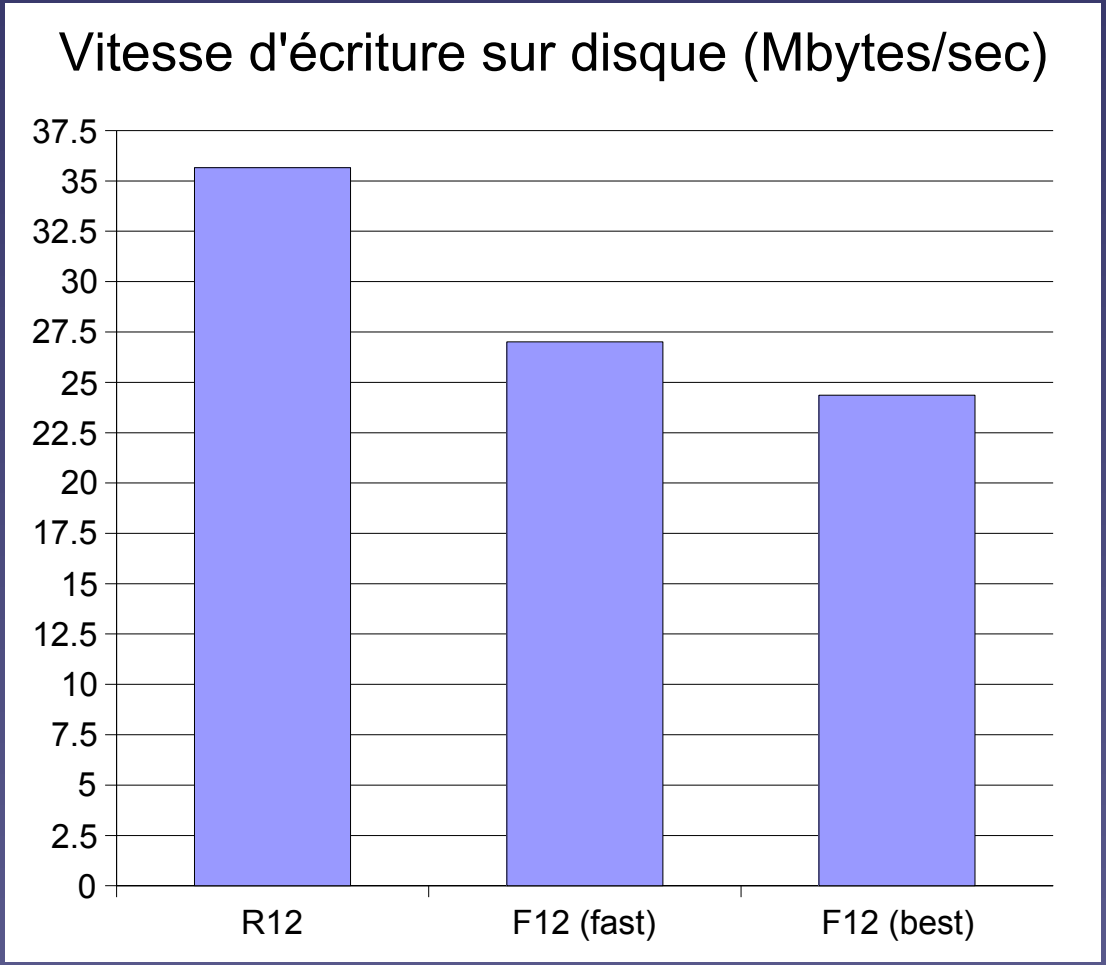




Lecture d'un fichier de 1,49 Gig (R12) généré avec
4 différents algorithmes de compression
(moyenne de 4 exécutions)

Traitement fait sur **phillips** (Pentium Xeon 2.66 GHz Debian Linux)

Vitesse d'écriture sur AZUR (GPFS)



Lorsque 100 processeurs écrivent en même temps sur le GPFS, l'activation de la compression donne un taux d'écriture qui est plus proche du maximum de bande passante du GPFS (2-3 Gb/s), résultant possiblement en un load plus balancé et un temps réel possiblement moindre.

Timings sur azur-test

time bemol_2006 -src dm* -ozsrt DM001.fst > DM001.out

where dm* is 100 files totalizing 196208 kbytes.

Input files	*Output file*	*Real*	*User*	*Sys*
Uncompressed	Uncompressed	49,37	19,98	2,99
Uncompressed	Compressed_fast	46,28	25,06	3,15
Uncompressed	Compressed_best	46,33	27,37	2,96

Utilisation des nouveaux algorithmes

- **fstcompress**
  - fstcompress -fstin source.fst -fstout dest.zfst -level fast/best
- **fstecr** (le binaire doit être linké avec librmn_rc008 ou plus récent)
  - datyp = 134 (6 + 128) pour les réels
  - datyp = 130 (2 + 128) ou 132 (4 + 128) pour les entiers
  - variable d'environnement FST_OPTIONS
  - `export FST_OPTIONS="DATATYPE_REMAP=1,134"`

Merci de votre attention