

# Calcul de structures d'ARN sur cartes graphiques

Dominique Lavenier

ENS Cachan Bretagne / IRISA

EPI INRIA Symbiose

# Plan de l'exposé

- Introduction
  - Parallélisme
  - Carte graphique
- L'algorithme UNAFold
  - présentation
  - Parallélisation
  - Implémentation sur cartes graphiques
- 2 exemples d'application
  - Recherche de micro ARN dans le génome du puceron
    - Collaboration : UMR BiO3P - INRA – Agrocampus Ouest
  - Évaluation statistique de la présence de structures secondaires dans une molécule d'ADN simple brin
    - Collaboration : UMR 6239 Génétique Immunothérapie Chimie & Cancer (GICC) - Tours

# Motivations

- Recherche de structures d'ARN
  - Algorithmes complexes et coûteux en temps
  - Besoins d'analyses à grande échelle (génomique)
- Les performances des microprocesseurs *mono-core* stagnent
  - Fréquence des processeurs bloquée depuis quelques années
  - Apparition des multi-cœurs (duo, quad, ...)
    - Les performances ne suivent pas l'évolution des masses de données génomiques
- Parallélisme massif
  - La seule solution pour réduire significativement les temps de calcul

# Parallélisme : plusieurs solutions

- Grille de calcul
  - Connexion via internet de plusieurs centre de calcul pour participer à un traitement
  - Parallélisme à gros grain
  - Distribution de tâches indépendantes
- Cluster (plate forme bioinfo GenOuest)
  - La solution la plus utilisée à ce jour
  - Parallélisme à gros grain
  - Possibilité de faire coopérer les nœuds pour un même traitement

# Parallélisme : plusieurs solutions

- Serveur multi cœurs
  - Plusieurs processeurs connectés à la même mémoire
  - Parallélisme à grain moyen
- Accélérateurs matériels
  - Carte dédiée à un calcul
  - Carte graphique
  - Parallélisme à grain fin

# Les cartes graphiques

- Architectures hautement parallèle
  - NVIDIA GTX 280 → 240 processeurs (1.3 GHz)
- Coût modeste
  - > 500 €
- Se programme facilement
  - Langage CUDA
  - OpenCL (en cours de standardisation)
- Installation aisée
  - Équipement immédiat des laboratoires de biologie
  - Pas de maintenance



# Le concept GP-GPU

- General-Purpose computation on Graphics Processing Units
- Concept né avec la flexibilité de programmation des cartes graphiques
- Déportation des calculs coûteux sur la carte graphique
- Tous les domaines applicatifs sont concernés
- En savoir plus : <http://gpgpu.org>



# Défis GP-GPU

- Transformer un programme écrit pour des ordinateurs standard (programmation séquentielle) en en programme massivement parallèle
- Limite :
  - loi d'Amdahl :
    - $S + P$  le temps d'exécution total d'un programme
      - $S$  est la fraction purement séquentielle
      - $P$  est la fraction parallélisable
    - Accélération max =  $(S+P) / S$
  - Temps de transfert des données vers la carte

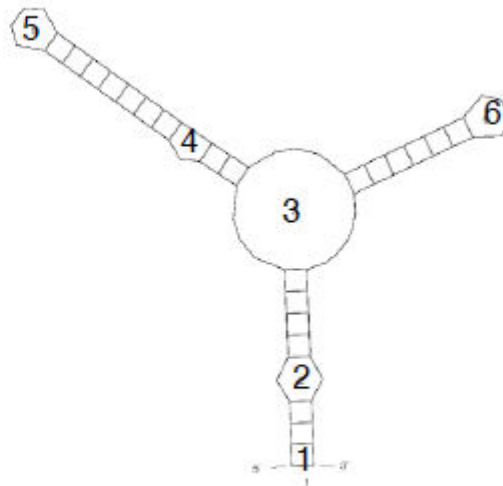
# UNAFold

- The UNAFold software package is an integrated collection of programs that simulate folding, hybridization, and melting pathways for one or two single-stranded nucleic acid sequences
- UNAFold = Unified Nucleic Acid Folding
- Folding algorithm  $\rightarrow$  *hybrid-ss-min* function



# hybrid-ss-min function

- Calcul de la structure la plus stable
  - En entrée : la séquence d'ARN
  - En sortie : structure + score



```
AAAAAAGGGAAAAGAACAAGGAGACUCUUCUCCUUUUUCAAGGAAGAGGAGACUCUUUCAAAAUCCUCUUUU  
(((.((((.(...(((.(((((((.....))))))))))....((((((.....)))))).....))))).))) (-24.5)
```

# Algorithme

$$Q'_{i,j} = \begin{cases} \min \left\{ \begin{array}{l} Eh(i, j) \\ Es(i, j) + Q'_{i+1, j-1} \\ \min_{k, l \in ]i; j[^2} Ei(i, j, k, l) + Q'_{k, l} \\ QM_{i+1, j-1} \end{array} \right\} & \text{if pair } i \cdot j \text{ is allowed} \\ \infty & \text{if pair } i \cdot j \text{ is not allowed} \end{cases} \quad (1)$$

$$QM_{i,j} = \min_{i < k < j} (Q_{i,k} + Q_{k+1,j}) \quad (2)$$

$$Q_{i,j} = \min \{ QM_{i,j}, \min(Q_{i+1,j}, Q_{i,j-1}), Q'_{i,j} \} \quad (3)$$

$Eh(i, j)$ ,  $Ei(i, j, k, l)$  and  $Es(i, j)$  are respectively the energies of :

- $Eh(i, j)$ : a hairpin loop closed by the pair  $i \cdot j$ .
- $Ei(i, j, k, l)$ : an interior loop formed by the two base pairs  $i \cdot j, k \cdot l$ .
- $Es(i, j)$ : two stacked base pairs  $i \cdot j$  and  $(i + 1) \cdot (j - 1)$ .

# Complexité

- N = taille de la séquence d'ARN
- T = temps d'exécution

$$T = f(N^3)$$

- 99 % du code est parallélisable

# Parallélisation

- Thèse de G. Rizk
- Algorithme difficile à paralléliser
  - Plusieurs mois de travail
  - Maîtrise de l'architecture des GPU
  - Première parallélisation « efficace »
- Gain sur UNAFold
  - Accélération : x10 à x25
  - Résultats rigoureusement identiques



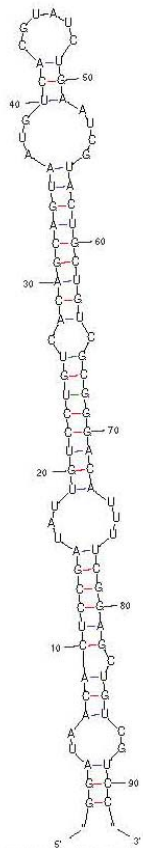
# 2 exemples d'applications

- Recherche de micro ARN dans le génome du puceron
  - Collaboration : UMR BiO3P - INRA – Agrocampus Ouest
  - D. Tagu, S. Jaubert, F. Legeai
- Évaluation statistique de la présence de structures secondaires dans une molécule d'ADN simple brin
  - Collaboration : UMR 6239 Génétique Immunothérapie Chimie & Cancer (GICC) – Tours
  - Y. Bigot, J. Cambefort

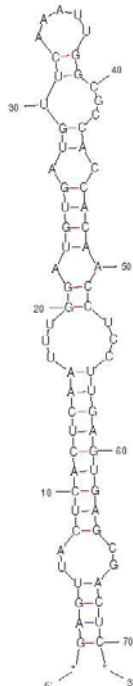
# Détection de microARN

- Les questions :
  - Quels sont les microARN présents dans le génome du puceron ?
  - Comment les identifier (vite et bien) ?
- La réponse bio-informatique :
  - Analyse “in silico” du génome du puceron pour sélectionner les meilleurs candidats potentiels
    - Choix d’algorithmes performants → UNAFold (bien)
    - Usage de cartes graphiques (vite)

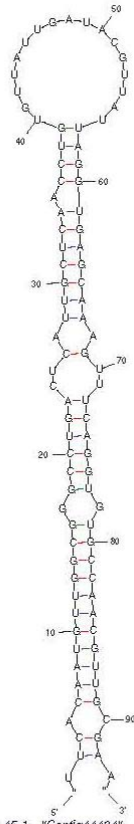
# Une structure générique mais beaucoup de variations



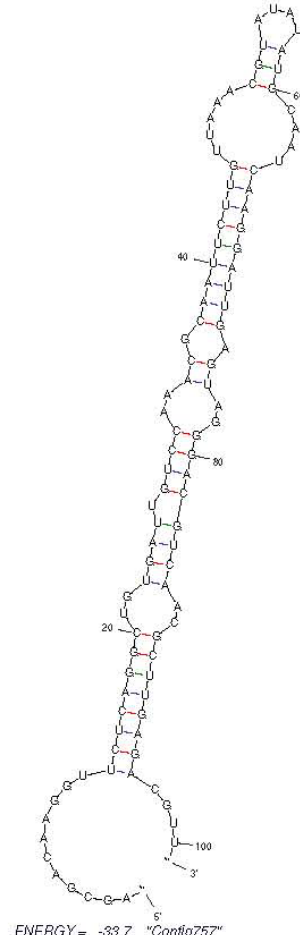
ENERGY = -48.3 "Contig27552"



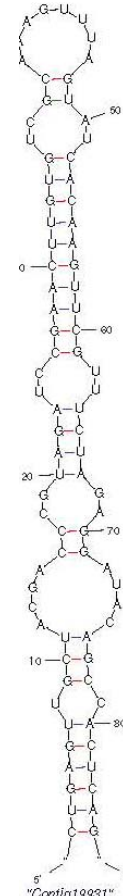
ENERGY = -25.8 "Contig22861"



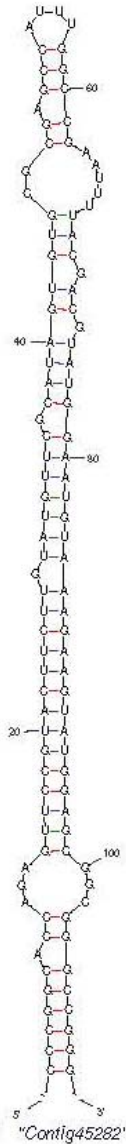
ENERGY = -45.1 "Contig44424"



ENERGY = -33.7 "Contig757"

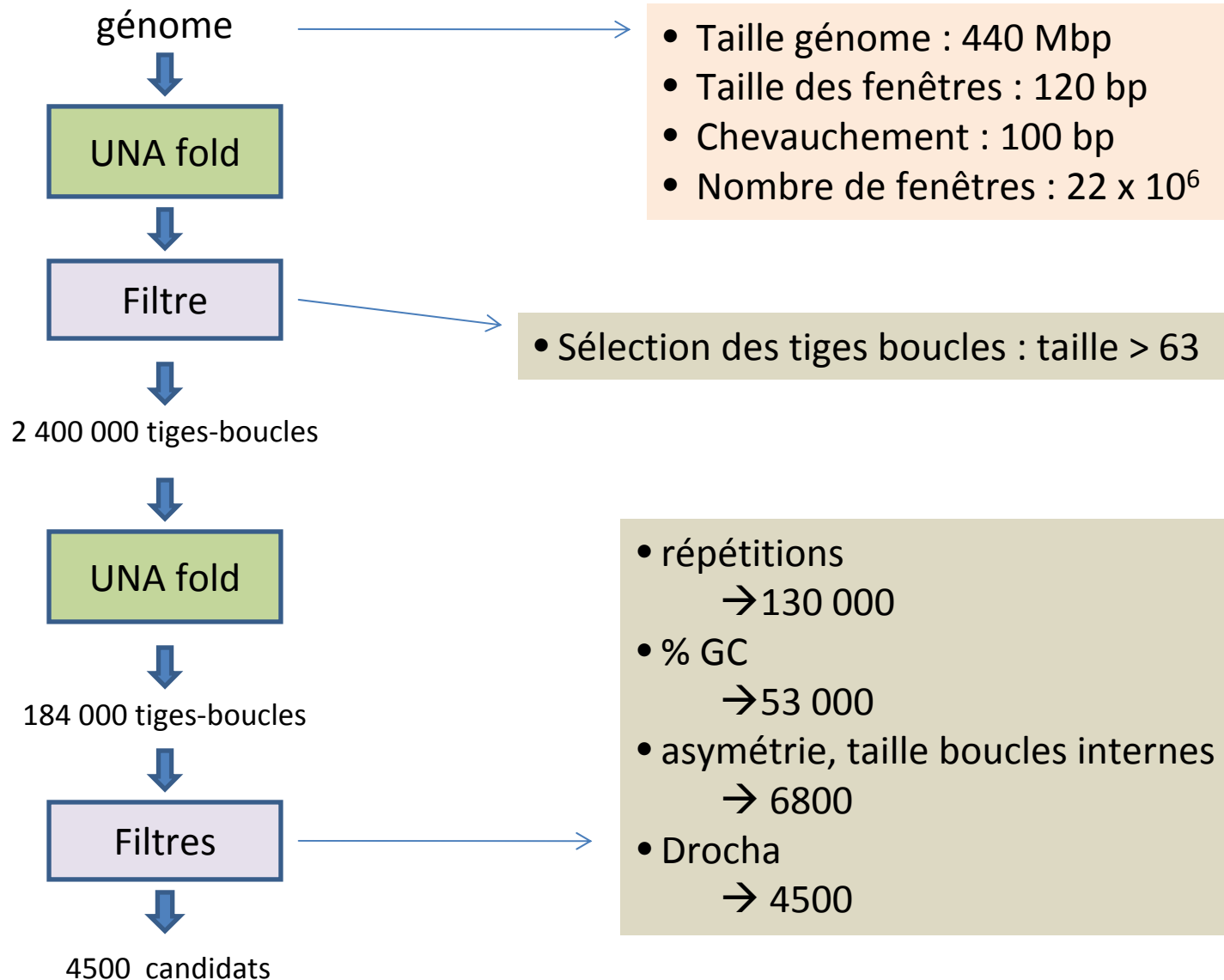


"Contig19831"

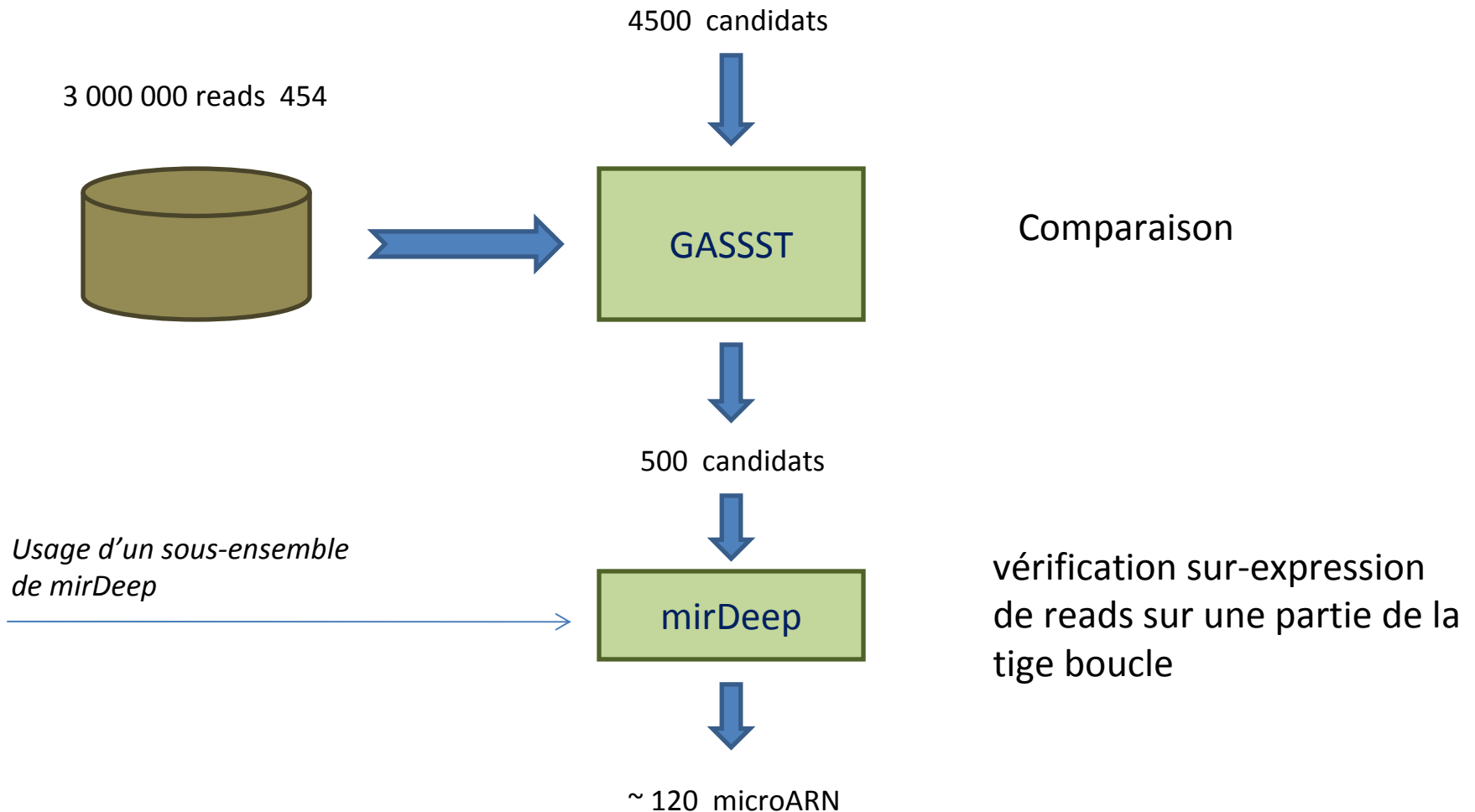


ENERGY = -52.8 "Contig45282"

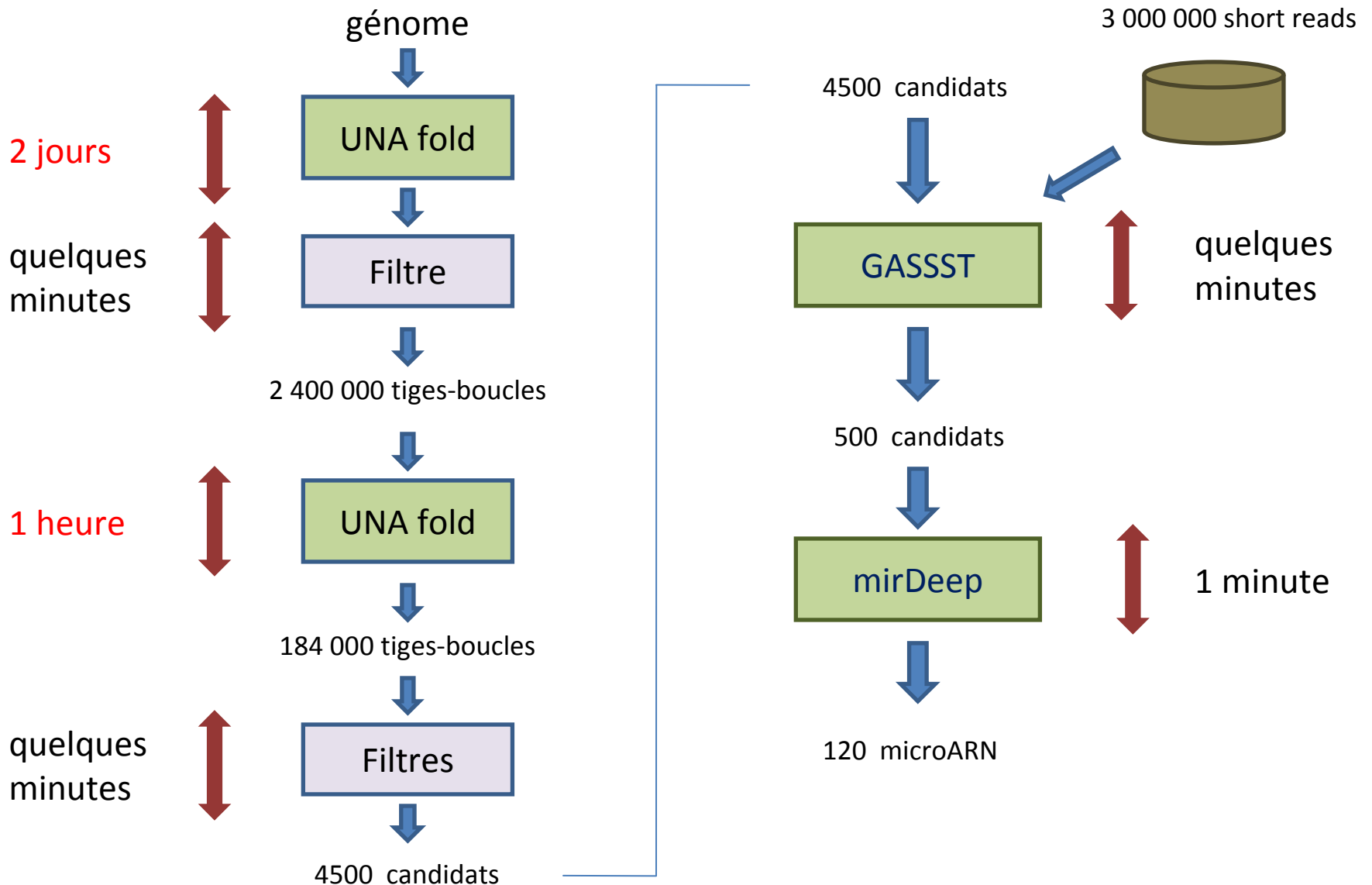
# Pipeline de détection (1)



# Pipeline de détection (2)



# Temps de calcul



# Accélération sur GPU

- $T_{\text{seq}} = T_{\text{UNAFold}} + T_{\text{filtres}}$   
 $= 48 \times 60 + 30$   
 $= 2880 + 30 = 2910$  minutes

- $T_{\text{par}} = T_{\text{UNAFold}} / 12 + T_{\text{filtres}}$   
 $= 240 + 30 = 270$

Accélération UNAFold



- Accélération globale :  $T_{\text{seq}} / T_{\text{par}} \approx 10$   
– 2 jours  $\rightarrow$  4h30

# Évaluation statistique de la présence de structures secondaires dans une molécule d'ADN simple brin

1. Séquence d'ADN simple brin

2. Prédiction de structures secondaires intrabrin d'ADN : UNAFold.

2 bis. Génération de N (1000 ou 10000) séquences aléatoires en conservant le % de bases : ShuffleSeq.

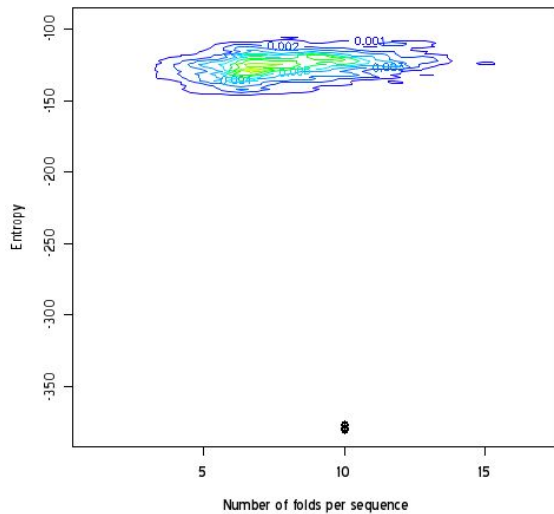
3. Extraction de critère de stabilité (nombre de repliements et entropies).

4. Evaluation statistique des résultats, et production d'un graphique d'ellipses de densité non-paramétriques.

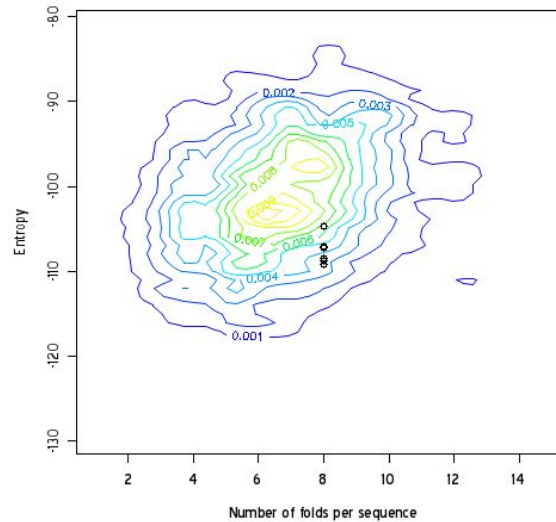


# Résultats du pipeline

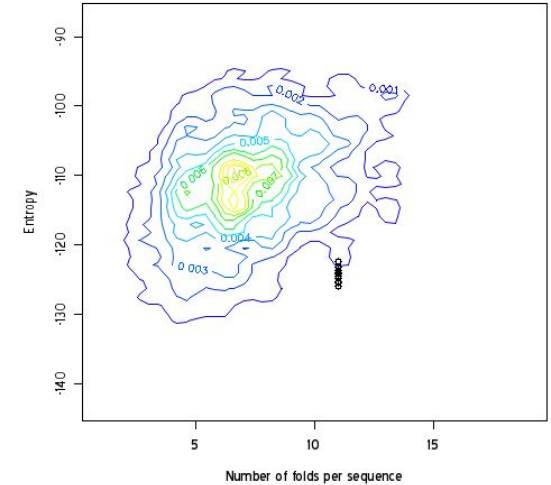
(pour 1000 séquences aléatoires générées)



Paris (ITmD34E-TLE)  
Structuré

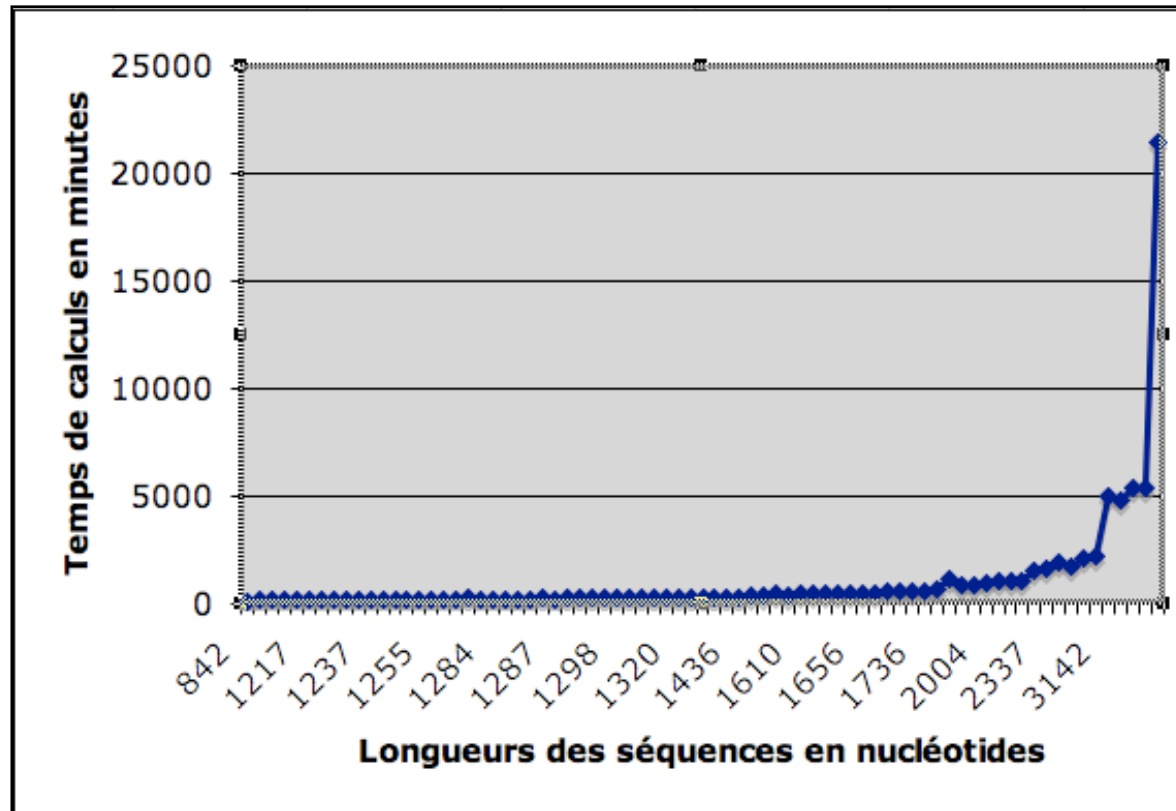


AcNFummar1  
(ITmD34D-MLE)  
Non structuré



AcNDemar1.2  
(ITmD34D-MLE)  
Indéterminé  
Augmenter le nombre  
de séquences aléatoires

# Temps de calcul



Temps de calcul en fonction de la taille des séquences, pour 1000 séquences aléatoires générées, avec un processeur Dual-Core Intel Xeon 2×2.66 Ghz.

# Pipeline

Séquence ADN  
simple brin  
1164 < taille 5259

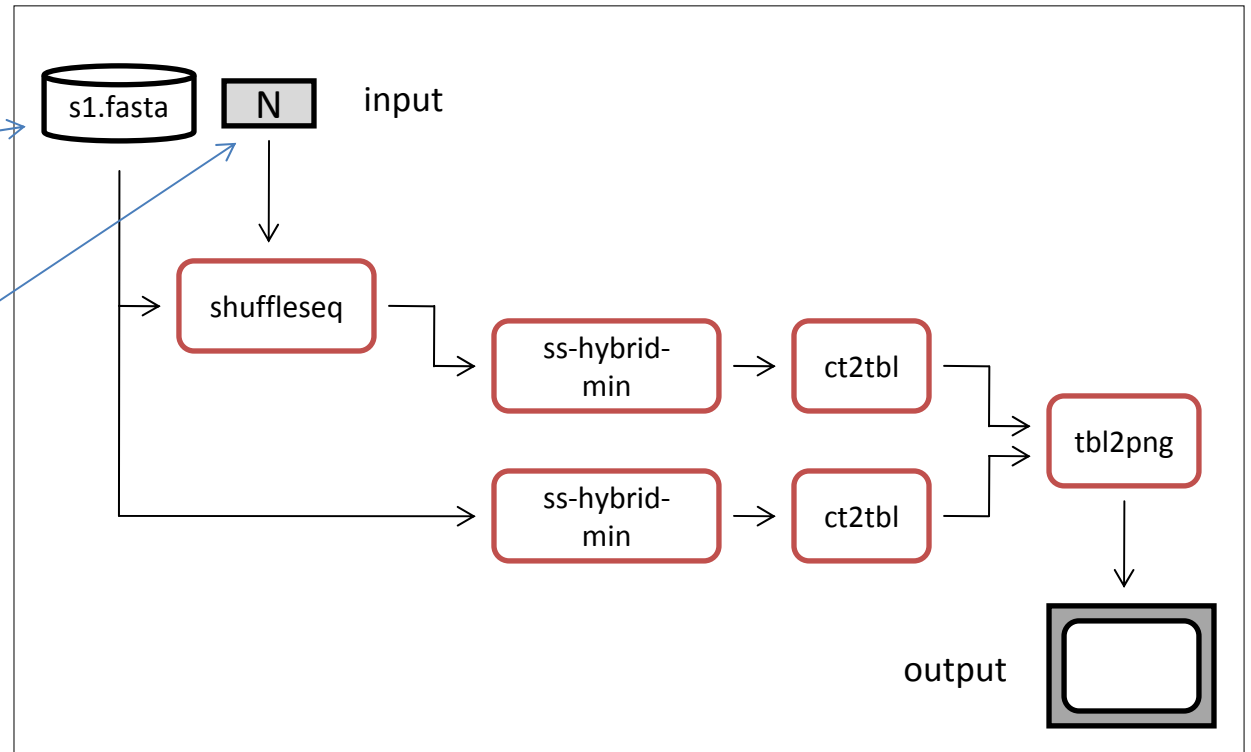
1000 < N 100 000

Temps de calcul

N=1000

5259 → 15 jours

(Dual-Core Intel Xeon 2.66Ghz.)



# Accélération sur GPU

- Plateforme :
  - Dual-Core Intel Xeon 2.66Ghz
  - 2 cartes TESLA (version pro des cartes graphiques)
- Accélération obtenue :
  - de x30 à x50
  - Plus la taille des séquences est grande, meilleure est l'accélération
    - 2 semaines → 7 heures (x50)
    - 3 heures → 6 minutes (x30)
- Equipement du labo
  - Installation validée sur une carte NVIDIA disponible
  - Achat en cours d'une carte graphique de dernière génération

# Conclusion

- GPU = technologie abordable
  - Coût réduit
  - Installation aisée
  - Pérennité : standard en cours → OpenCL
- Mise en œuvre des algorithmes
  - Difficile → extraction d'un parallélisme massif
- Programmes Parallélisés sur GPU
  - Smith & Waterman
  - PLAST
  - HMMER
  - Docking (Piper)
  - ...