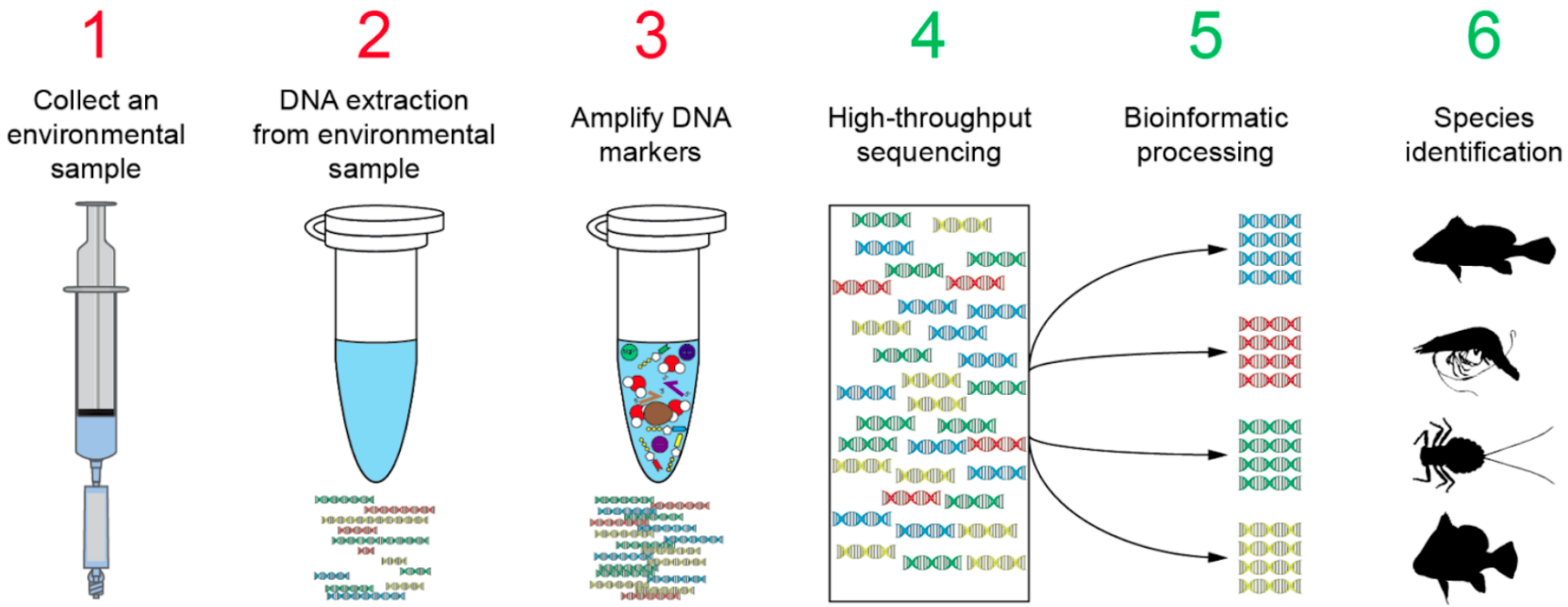
Inria

JCAD 2021

Passage à l'échelle d'un algorithme de Randomized SVD pour la caractérisation du vivant par méthode de positionnement multidimensionnel

E. Agullo, O. Coulaud, M. Faverge, A. Franc, J.M. Frigerio, F. Pruvost

Metabarcoding



- Généralement : $\approx$ 10 000 individus
- Challenge : 1 000 000 individus $\Rightarrow$ **calcul parallèle distribué**
- Données : Inrae, diatomées (algues) du lac Léman

Positionnement multidimensionnel - MDS

Matrice de distance D

$$\begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{n,1} & d_{n,2} & \cdots & d_{n,n} \end{pmatrix}$$

Projection sur un espace Euclidien $r \ll n$

$\longrightarrow$

Conservation des distances

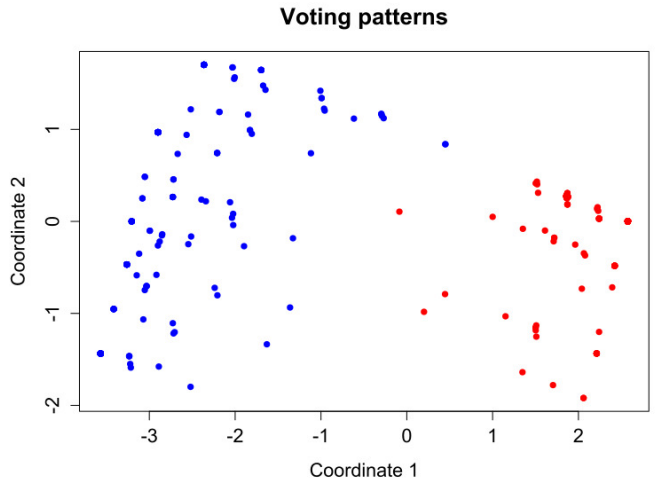
Nuage de points X

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,r} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,r} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,r} \end{pmatrix}$$

Visualisation de X en 2 ou 3 dimensions

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,r} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,r} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,r} \end{pmatrix}$$

X représenté sur un plan



Inria

Positionnement multidimensionnel - MDS

Matrice de distance D

$$\begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{n,1} & d_{n,2} & \cdots & d_{n,n} \end{pmatrix}$$

Projection sur un espace Euclidien $r \ll n$

$\longrightarrow$

Conservation des distances

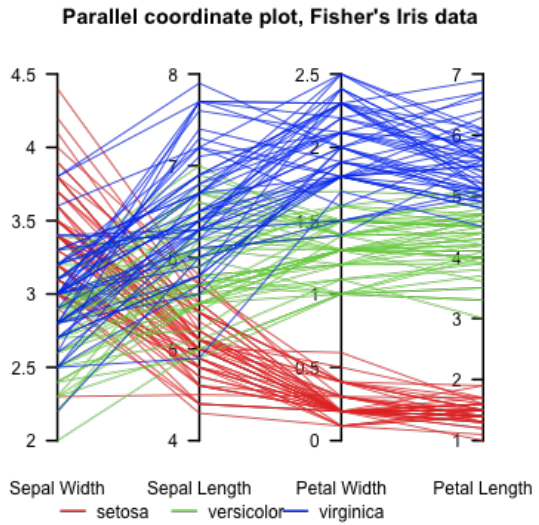
Nuage de points X

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,r} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,r} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,r} \end{pmatrix}$$

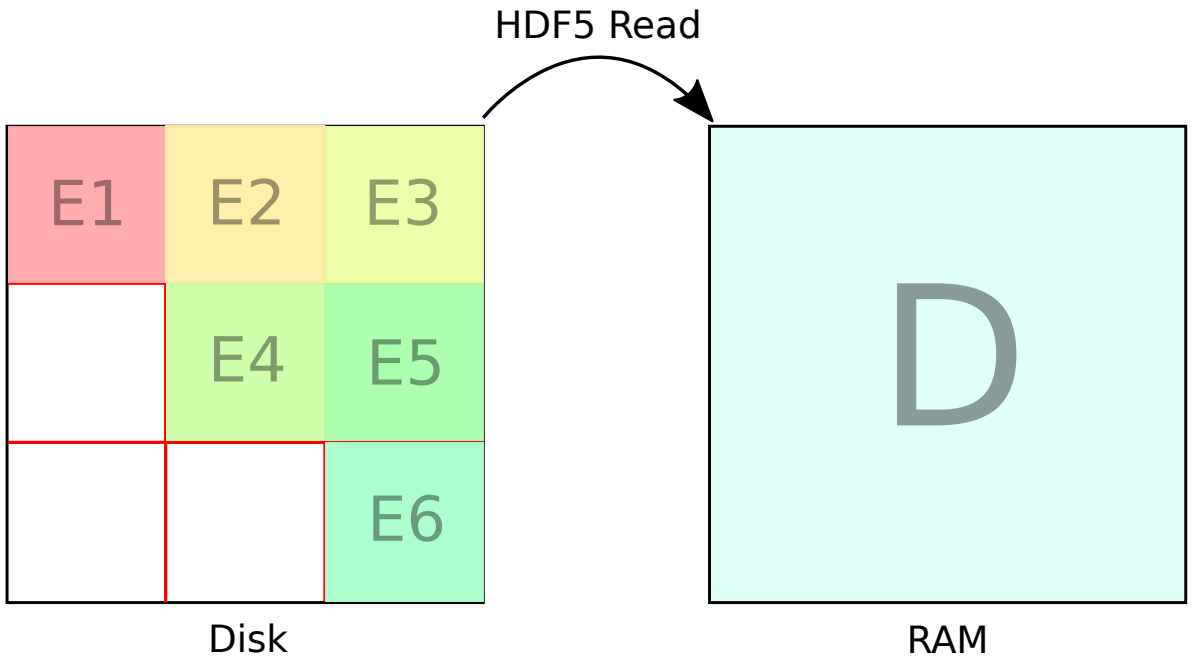
Coordonnées parallèles

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,r} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,r} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,r} \end{pmatrix}$$

1 courbe = 1 individu

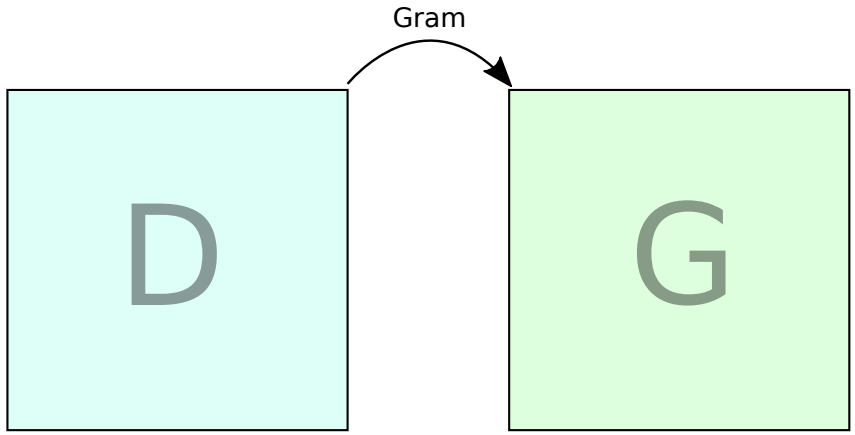


1) Lecture matrice de distance D



Plusieurs échantillons, fichiers HDF5

2) Matrice de Gram (double recentrage) $D \rightarrow G$

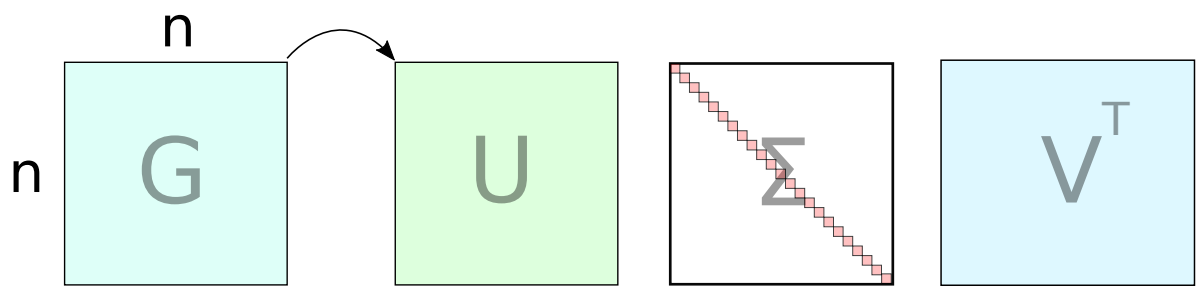


$$d_{i.}^2 = \frac{1}{n} \sum_j d_{ij}^2$$

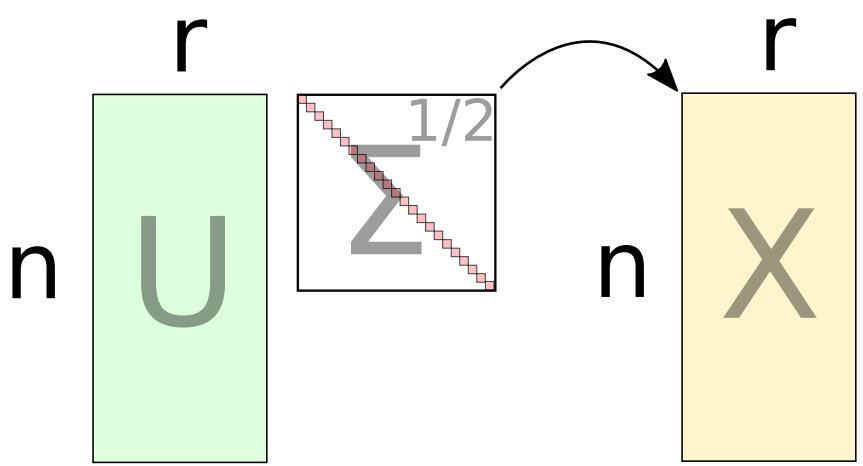
$$d_{..}^2 = \frac{1}{n^2} \sum_{i,j} d_{ij}^2$$

$$G_{i,j} = -\frac{1}{2}(d_{ij}^2 - d_{i.}^2 - d_{.j}^2 + d_{..}^2)$$

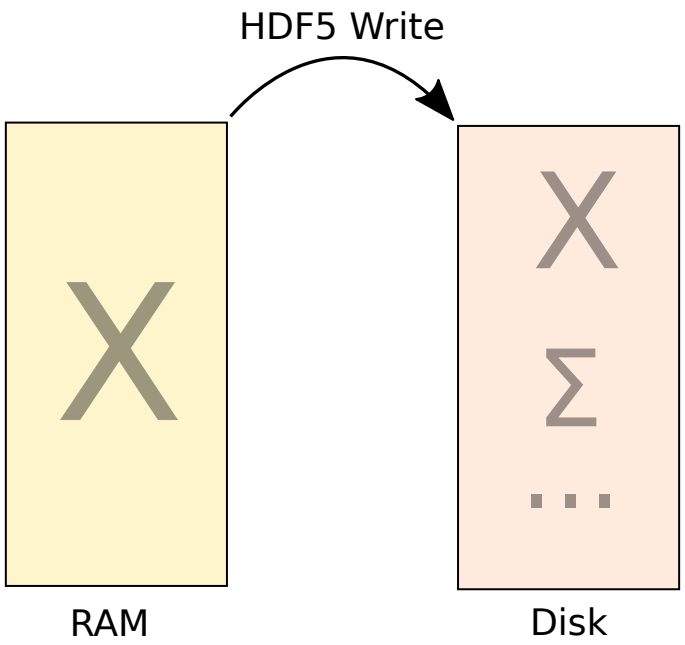
3) Singular Value Decomposition (SVD) $G = U\Sigma V^T$



4) Nuage de point $X = U\Sigma^{1/2}$



5) Ecriture HDF5 matrice X, Σ, metadata ...

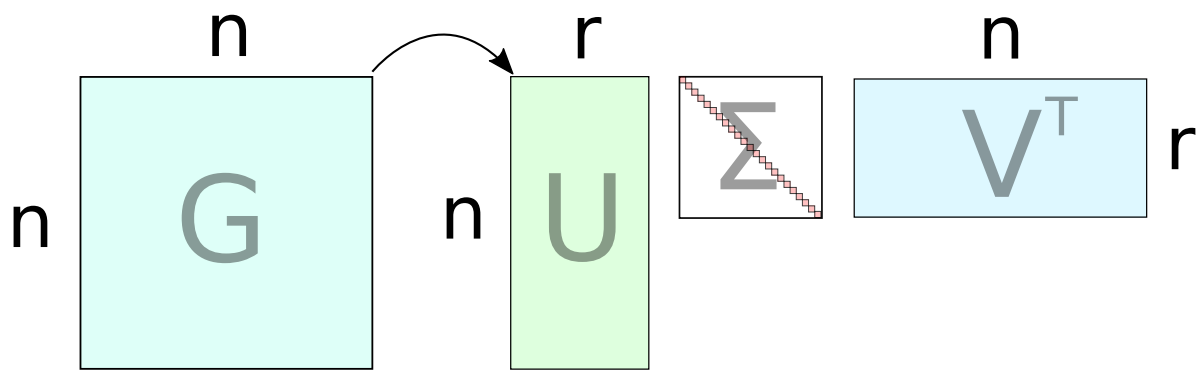


1. Lecture HDF5, assembler D
2. Transformation de Gram, $D \rightarrow G$
3. SVD sur G, $G = U\Sigma V^T$
4. Nuage de points, $X = U\Sigma^{1/2}$
5. Ecriture HDF5 de X

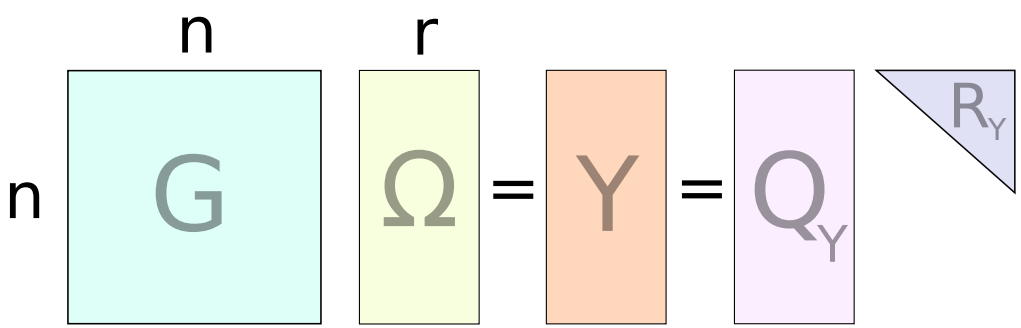
- Faisable en Python, $n < 10000$
- Capable de traiter des problèmes de taille n = 1 million ?
  - > Non, la SVD est trop coûteuse, $O(n^3)$!
- En pratique G de rang faible, $r \ll n$ $(r = 10^3, n = 10^6)$
  - > Approcher la SVD de G par Randomized SVD, $O(n^2 \times r)$

Inria

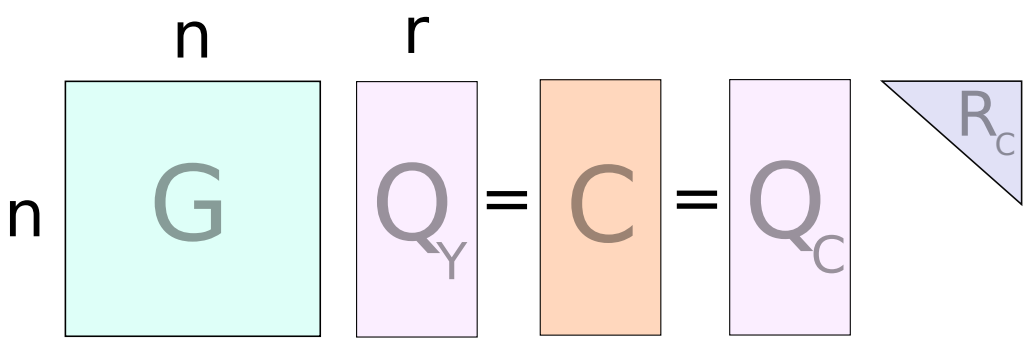
Lorsque rang de G est $r \ll n$, la SVD peut être tronquée



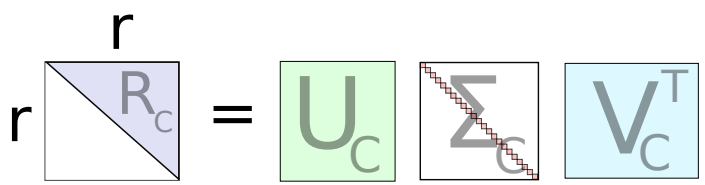
1. Génération d'une matrice aléatoire Gaussienne : Ω
2. Produit de matrices : $Y = G\,\Omega$
3. Décomposition QR : $Y = Q_Y R_Y$

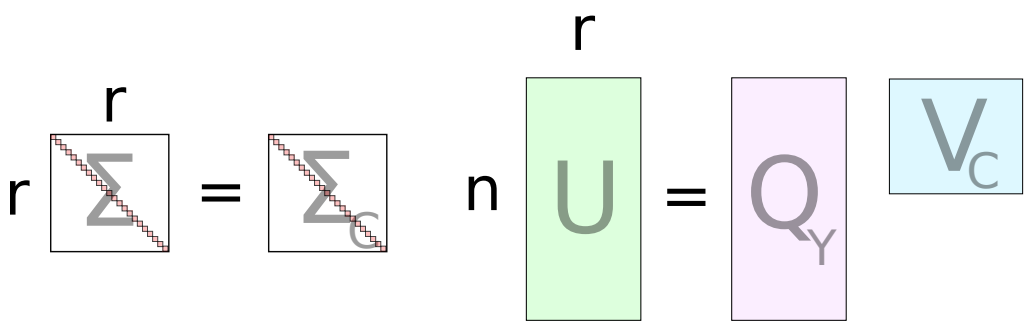


4. Produit de matrices : $C = G^T Q_Y = G\ Q_Y$, G symétrique
5. Décomposition QR : $C = Q_C R_C$



6. Décomposition SVD : $R_C = U_C \Sigma_C V_C^T$, $\Sigma = \Sigma_c$
7. Produit de matrices : $U = Q_Y V_C$, $V^T = U_C^T Q_C^T$

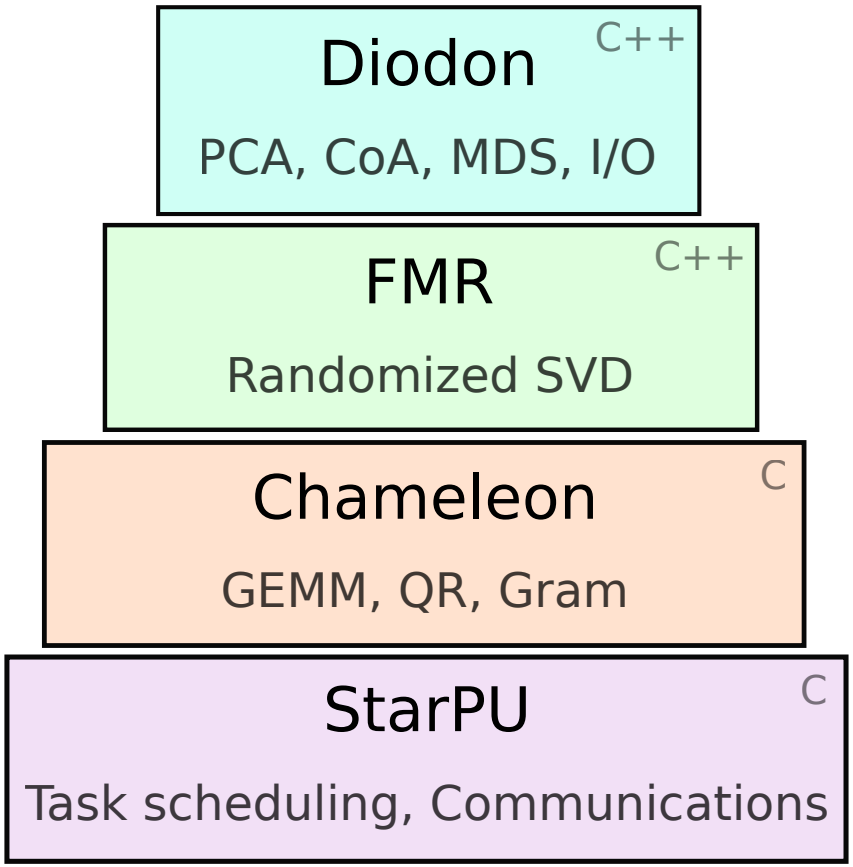




1. GEMM : $Y = G\,\Omega$, $O(n^2 \times r)$
2. QR : $Y = Q_Y R_Y$, $O(n \times r^2)$
3. GEMM $C = G\,Q$, $O(n^2 \times r)$
4. QR : $C = Q_C R_C$, $O(n \times r^2)$
5. SVD : $R_C = U_C \Sigma_C V_C^T$, $O(r^3)$
6. GEMM : $U = Q_Y V_C$, $O(n \times r^2)$

- Complexité dominée par les produits de matrices $O(n^2 \times r)$
- GEMM : très parallélisables !
- QR : raisonnablement parallélisables
- SVD : petit système, calculable sur un noeud

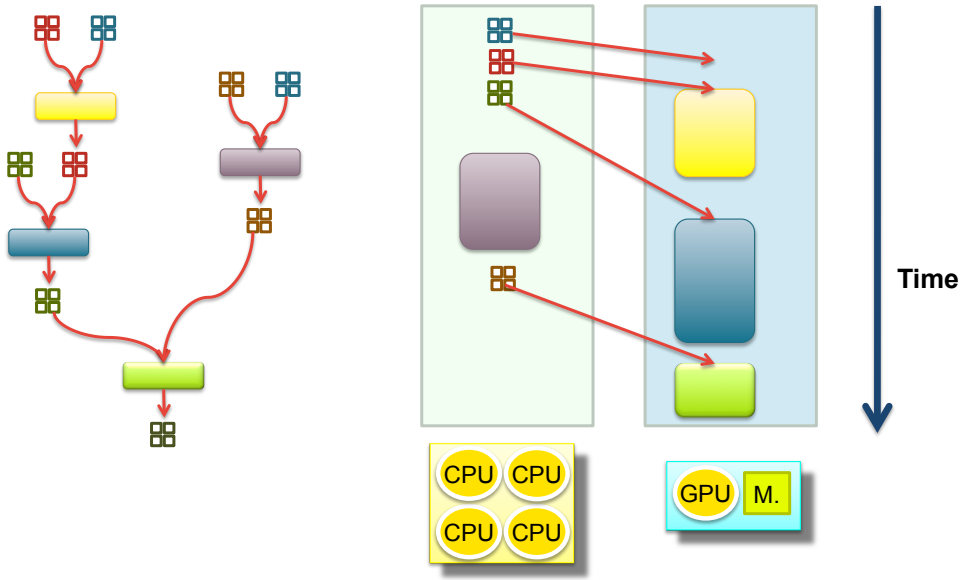
Pile logicielle d'algèbre linéaire parallèle



Inria

Le moteur d'exécution **StarPU**

- Soumission d'un graphe de tâches (DAG)
- Gestion des dépendances et des transferts de données (MPI/CUDA)
- Différentes stratégies d'ordonnancement
- Portabilité des performances

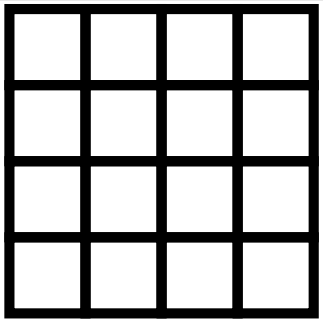


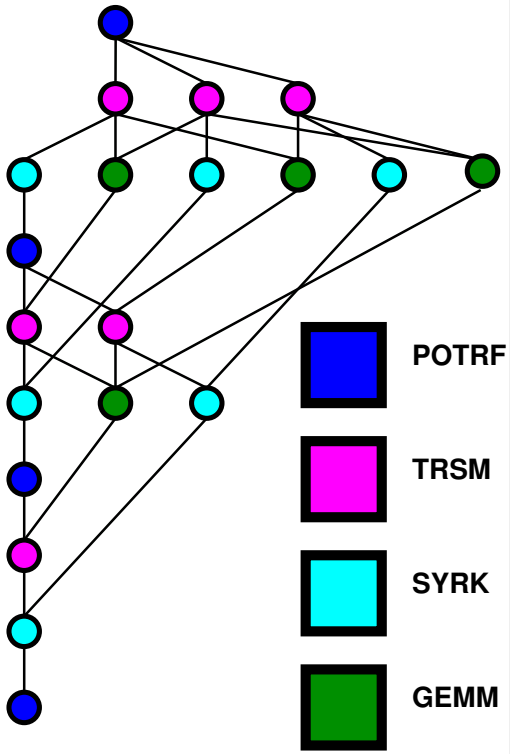
Ex. Sequential Task-Based Cholesky

Solveur d'algèbre linéaire dense **Chameleon**

- format en tuiles → noyaux (CPU, GPU)
- graphe de tâches → moteur d'exécution

```
for (j = 0; j < N; j++) {
    POTRF (RW,A[j][j]);
  for (i = j+1; i < N; i++)
    TRSM (RW,A[i][j], R,A[j][j]);
  for (i = j+1; i < N; i++) {
    SYRK (RW,A[i][i], R,A[i][j]);
    for (k = j+1; k < i; k++)
      GEMM (RW,A[i][k],
            R,A[i][j], R,A[k][j]);
  }
}
__wait__();
```





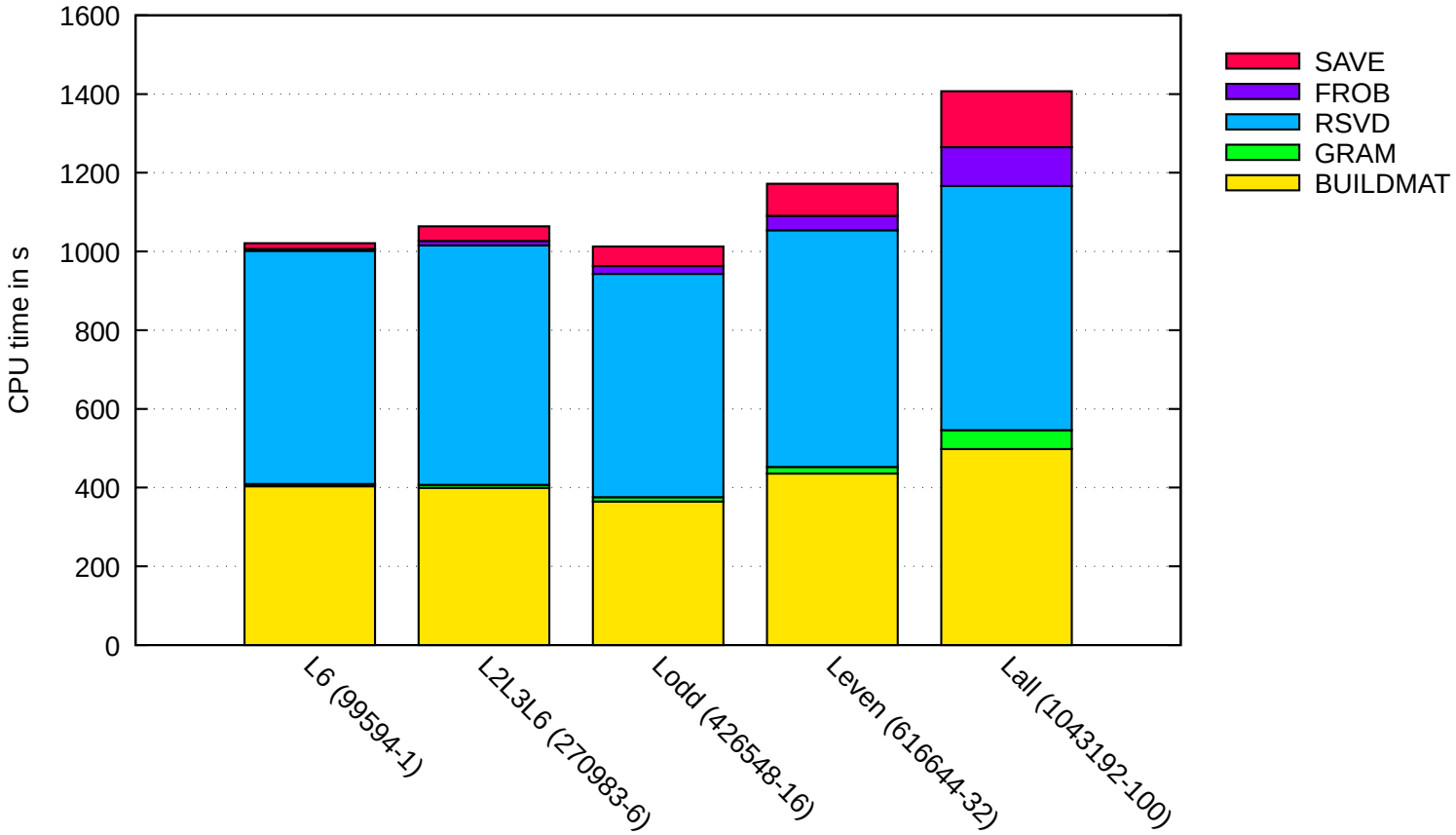
Inria

Résultats

Inria

Temps CPU de la MDS, $r = 10^4$, cas complexe à résoudre

La RSVD passe bien à l'échelle !

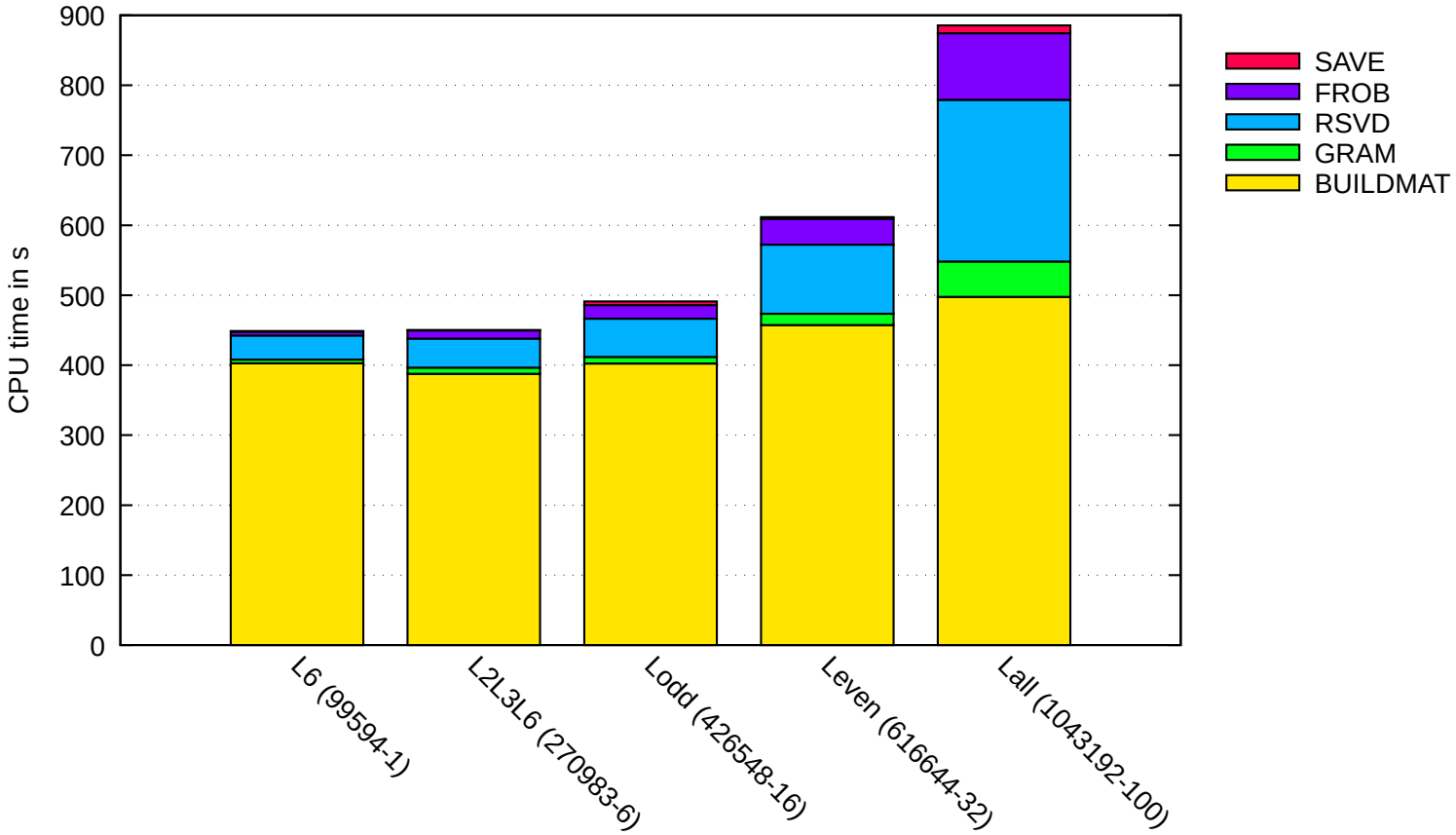


CINES Occigen, $n \in [10^5, 10^6]$, #Nodes $\in [1, 100]$, #CPUs $\in [24, 2400]$

Inria

Temps CPU de la MDS, $r = 10^3$, rang plus réaliste en pratique

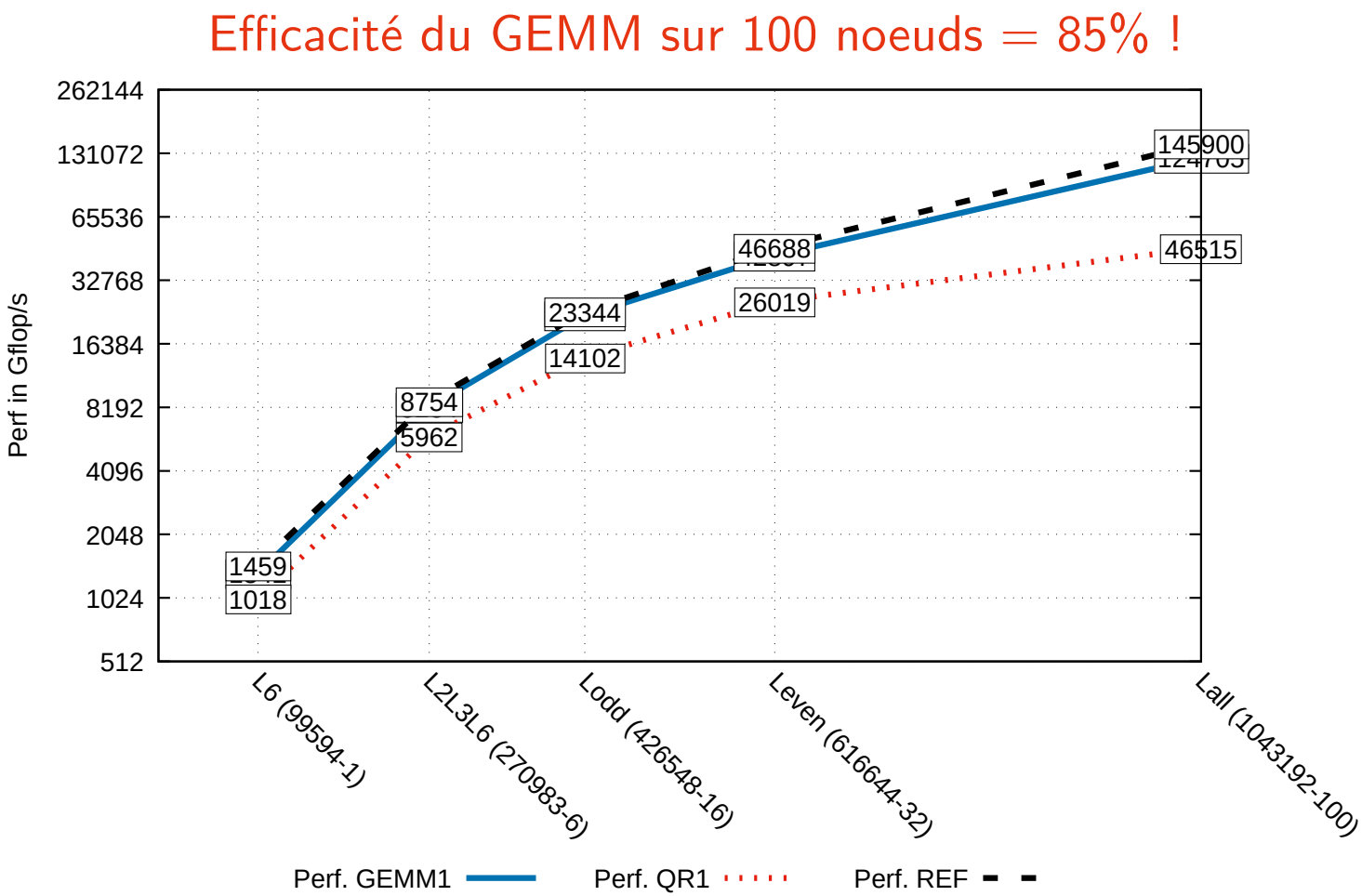
La MDS du problème 1 million se résout en moins de 15 min !



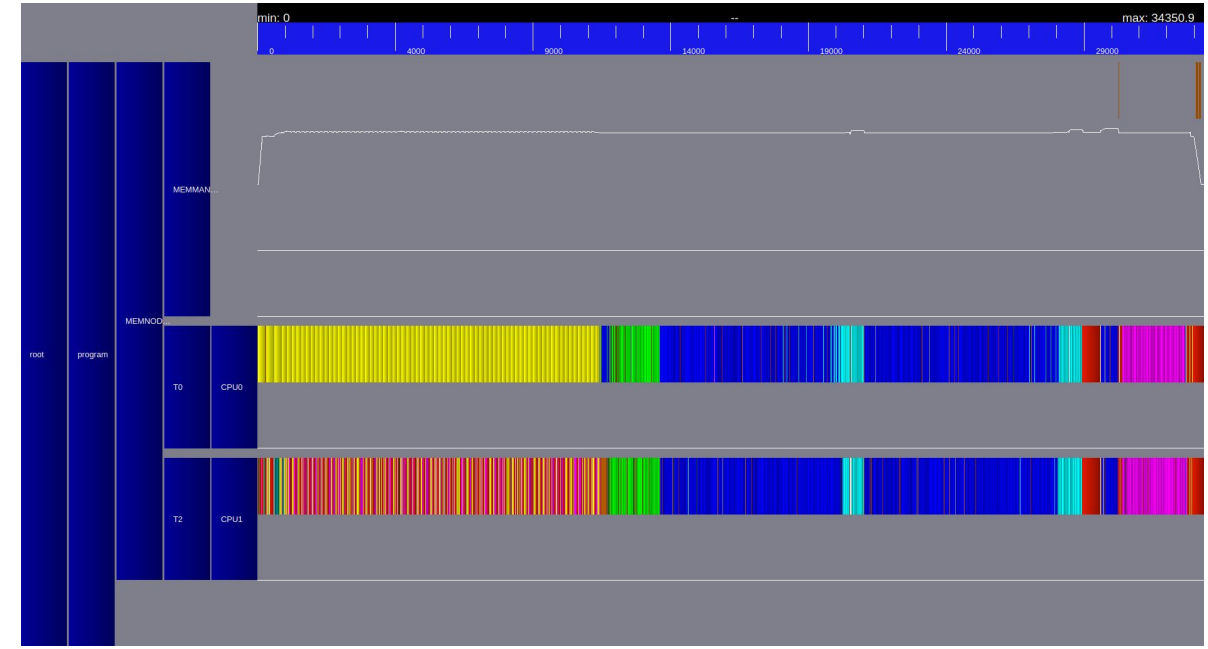
CINES Occigen, $n \in [10^5, 10^6]$, #Nodes $\in [1, 100]$, #CPUs $\in [24, 2400]$

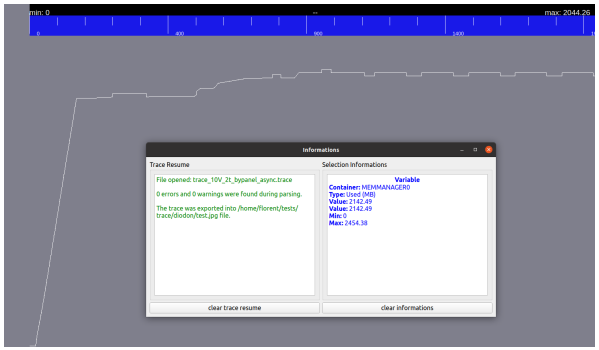
Inria

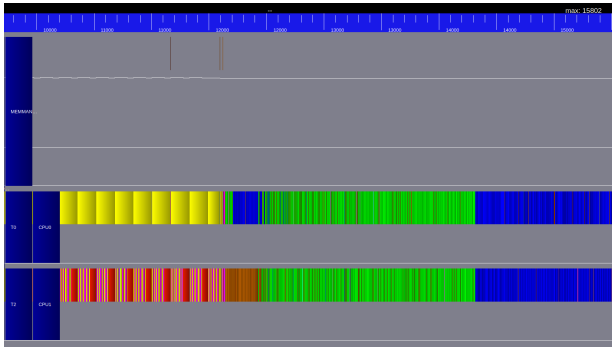
Performances en terme de GFlop/s

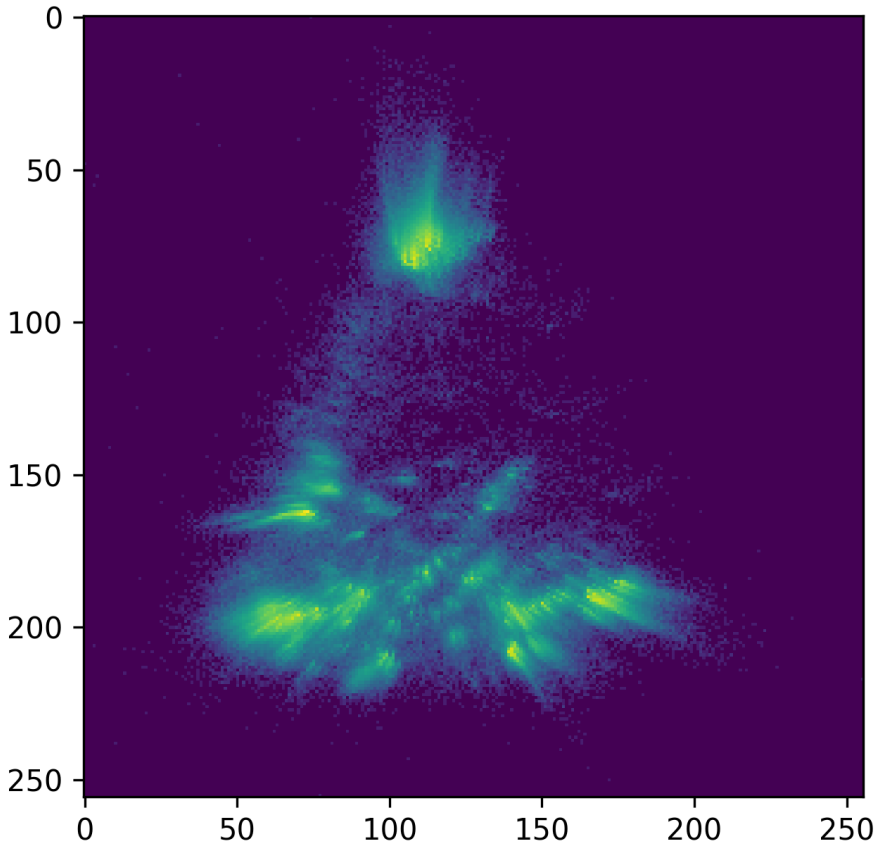


Traces d'exécution (StarPU, EZTrace, ViTE)









Représentation de la MDS sur 2 axes, $n = 10^6$

Conclusions

- Grâce à une pile logicielle d'algèbre linéaire mature nous avons pu résoudre le challenge de SVD (approchée) $n = 10^6$
- La MDS peut fournir des solutions de références à comparer avec des méthodes de "clustering" avec heuristiques moins coûteuses mais moins robustes (Swarm)
- Les temps de calculs deviennent moindres devant les I/O si la matrice est de rang très faible
- Principales difficultés :
  - > I/O, conversion formats HDF5 $\rightarrow$ tuiles Chameleon,
  - > contrôle de la consommation mémoire avec StarPU

Perspectives

- Mettre à disposition **Diodon** (MDS, PCA, ...) en licence libre
- Faciliter l'utilisation du logiciel sur supercalculateurs
  - > serveur Galaxy, Jupyter notebook, packaging GNU Guix
- Trouver de nouvelles applications SVD sur grandes données
  - > compression, problèmes inverses, POD, ...
- Réaliser des tests avec des GPUs
  - > expérimentations en cours sur IDRIS Jean Zay
- Réaliser un portage sur un système "big data"
  - > écosystèmes yarn/hdfs
  - > se comparer à Hadoop MapReduce et Spark
- Travailler sur une SVD exacte passant à l'échelle
  - > si r devient grand, l'étape de SVD sur R_C devient bloquante

Merci pour votre attention !