PLONGEMENTS DE TEXTES DE BREVETS ET OPTIMISATION DU CALCUL D'INDICATEURS

—

JCAD
14 DÉCEMBRE 2021

Ziad KACHOUH (OST)
Patrick BOUSQUET-MELOU (CRIANN)
Aymeric de CARFORT (OST)
Hermann WOEHREL (OST)
Jean-Marc DELTORN (UNISTRA)
Dominique GUELLEC (OST)

Logistique du projet

Déroulement :
- Projet mené d'octobre 2020 à juin 2021.

Projet coopératif :
- mené par une équipe de l'Observatore des Sciences et Techniques (OST), département du Hcéres (Haut Conseil pour l'Evaluation de la Recherche et de l'Enseignement Supérieur).
- impliquant un chercheur de l'Université de Strasbourg
- utilisation du supercalculateur Myria (du CRIANN)

Contexte et objectifs du projet

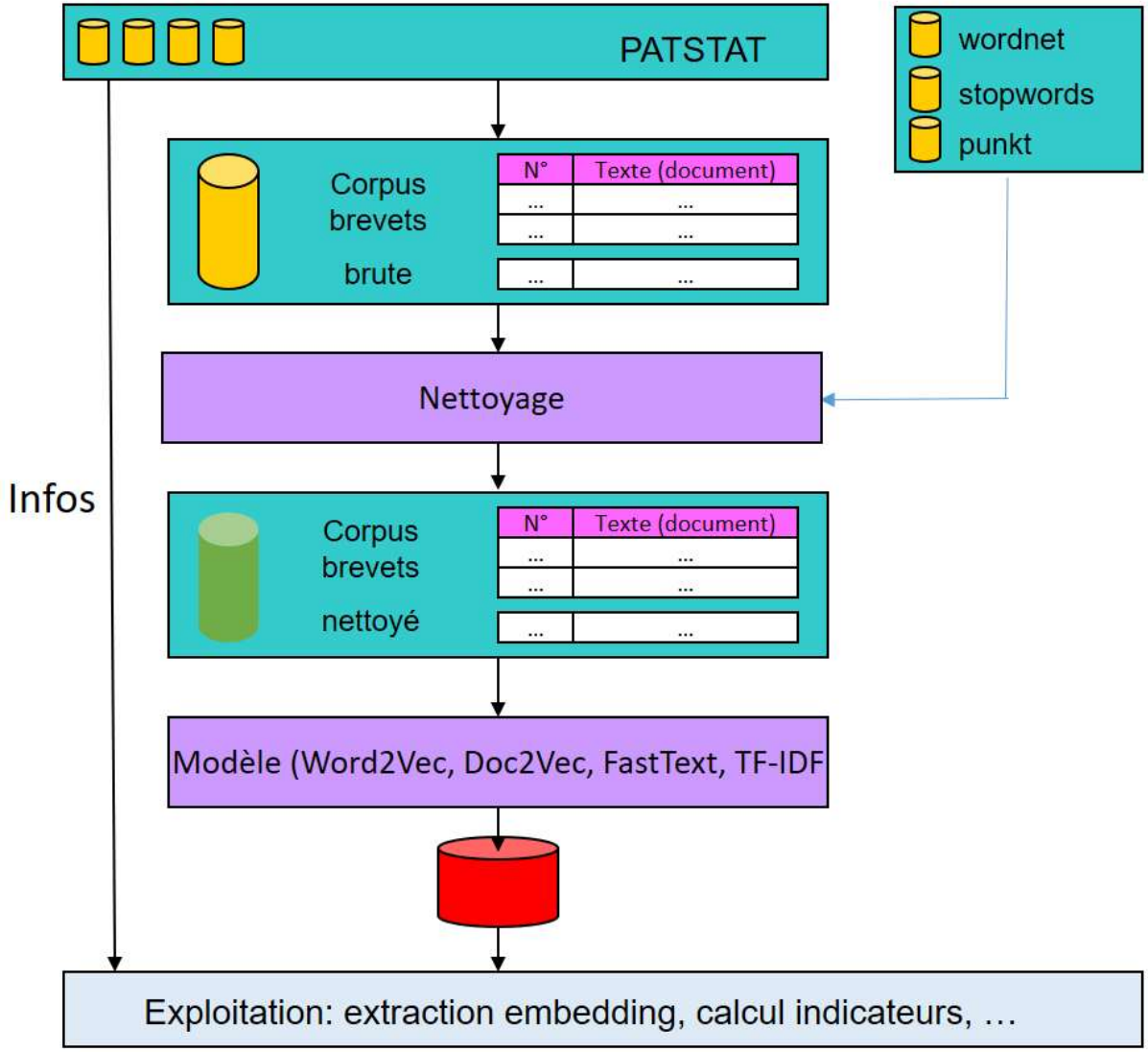
Les indicateurs actuels d'innovation fondés sur les brevets exploitent exclusivement les **métadonnées** (inventeurs, adresses, institutions, citations, classes etc.).
=> Il s'agit ici d'exploiter les **données textuelles**, qui reflètent le CONTENU des inventions.
Cela permettrait d'améliorer les indicateurs existants et de concevoir des indicateurs nouveaux, intégrant la **similarité**/ distance technique entre brevets => nouveauté, impact, filiation, clusters etc.

Objectifs du projet :

1) Explorer les opportunités et difficultés d'une analyse sémantique des brevets

2) Concevoir et calculer des indicateurs de similarité et « d'inventivité »

Séquence des opérations de traitement des données



Constitution du corpus

Extraction de Patstat, Brevets OEB 1978-2018 et USPTO 1970-2018
=> titre et résumé en anglais
=> métadonnées: statut légal du brevet (demande, délivré), catégorie technologique (CPC), citations

Nettoyage des données : le texte contient différents types d'éléments qui peuvent gêner la vectorisation : balises, ponctuation, fautes d'orthographe. Les mots les plus courants n'ont pas de sens intrinsèque (articles), ou sont trop polysémiques (« invention »). Selon les techniques utilisées : variations orthographiques (conjugaison, pluriel).
=> *Le nettoyage approprié dépend de l'usage des données résultantes*

Les corpora – statistiques globales

Tableau 1 : L'impact du nettoyage sur la taille des corpora		
	Corpus 1 – OEB	Corpus 2 – OEB-USPTO
Nombre de documents	3,3 millions	8,5 millions
Nombre initial de termes	363 millions	1 milliard
Nombre de termes différents	2,6 millions	4,6 millions
Après nettoyage : nombre de tokens total	211 millions *(lemmatisation, unigram)*	609 millions *(racinisation, unigram)*
Tokens distincts	412.000	504.000

Exemple de transformation de texte

Texte brut :

non isolated bidirectional dc-dc converter having improved stability

Nettoyage avec lemmatisation :

non isolate bidirectional converter improve stability

Nettoyage avec racinisation :

non isol bidirect convert improve stabil

Plongements lexicaux (« embeddings »)

Plongement lexical : méthode consistant à transformer un texte (mot, document) en vecteur.

Le méthodes de traitement automatique de la langue les plus récentes/performantes reposent sur des plongements.

Intérêt : un vecteur 1) est défini dans un espace doté d'une métrique (distance) ; 2) peut être manipulé avec des opérations algébriques (additions etc.) permettant des compositions de sens.

La difficulté : former des plongements qui reflètent fidèlement le sens des textes.

Il existe différentes méthodes de plongements.

Plongements lexicaux – word2vec et doc2vec

Un modèle (équivalent à un réseau de neurones à deux couches) est ajusté, avec des paramètres (coordonnées des vecteurs-mots) qui optimisent pour chaque mot sa prédiction par ses voisins (continuous bag of words, cbow).
Word2Vec est fondé sur le principe énoncé par le linguiste Firth dans les années 50, "a word is characterized by the company it keeps".
Doc2Vec est une méthode dérivée de Word2Vec ; elle consiste a intégré l'appartenance à un document particulier dans l'estimation des paramètres du modèle, permettant ainsi d'obtenir un plongement pour chaque document qui prend en compte non seulement les mots qui le composent individuellement, mais aussi la singularité de leur regroupement.

Ajuster les hyperparamètres

Hyperparamètres :
- nombre de dimensions du vecteur (de 12 à 300)
- taille de la fenêtre » (nombre de voisins pris en compte dans la prédiction)
- nombre d'itérations de l'entrainement

Nous avons procédé à de multiples tests : les résultats présentés ici correspondent à des valeurs sélectionnées.

Exemple de plongement d'un document

Texte brut d'un brevet

Processing methods and systems for assembling fuel cell perimeter gaskets A method and apparatus for making fuel cell components via a roll to roll process are described. Spaced apart apertures are cut in first and second gasket webs that each include adhesives. The first and second gasket webs are transported to a bonding station on conveyers. A membrane web that includes at least an electrolyte membrane is also transported to the bonding station. At the bonding station, a gasketed membrane web is formed by attaching the first and second gasket webs to the membrane web. The first gasket web is attached to a first surface of the membrane web via the adhesive layer of the first gasket web. The second gasket web is attached to a second surface of the membrane web via the adhesive layer of the second gasket web.

Le texte nettoyé du brevet

process systems assemble fuel_cell perimeter gasket apparatus make fuel_cell components via roll roll process describe space apart apertures cut gasket web adhesive gasket web transport bond station conveyers membrane web least electrolyte membrane also transport bond station bond station gasketed membrane web form attach gasket web membrane web gasket web attach surface membrane web via adhesive layer gasket web gasket web attach surface membrane web via adhesive layer gasket web

Le vecteur correspondant au texte du brevet de taille 12

[-0.8812195 1.3134036 0.6439331 -1.0924578 -1.2354546 2.6685338
0.6614424 0.05630265 -1.1950052 0.08640682 2.0768085 1.6230906]

Les plongements conservent les relations de synonymie

Synonymie :

```
# les mots les plus proches de "method"
model.wv.most_similar('method')
```

```
[('methodfor', 0.9090000987052917),
 ('submethod', 0.8949490189552307),
 ('techniqu', 0.8883976340293884),
 ('methodof', 0.8837336301803589),
 ('methodolog', 0.8787580728530884),
 ('ofprepar', 0.852911114692688),
 ('nethod', 0.846514105796814),
 ('use', 0.8348050117492676),
 ('ofmanufactur', 0.8313615918159485),
 ('repar', 0.8241061568260193)]
```

```
# les mots les plus matrix de "hormon"
model.wv.most_similar('hormon')
```

```
[('ofhormon', 0.9367474913597107),
 ('antihormon', 0.8957105875015259),
 ('parathormon', 0.8905232548713684),
 ('pancreocymincholecystokinin', 0.882892549
 ('adrenocorticotroph', 0.8770840167999268),
 ('proneurotrophin', 0.8762346506118774),
 ('interleukin', 0.8706635236740112),
 ('cardiotrophin', 0.8629087805747986),
 ('cholecystokinin', 0.8602062463760376),
 ('adrenocorticotrop', 0.8596653938293457)]
```

Les plongements préservent la compositionnalité du sens des mots

Combinaison de deux mots :

	inkjet	ink	inkdrop	jet	inkwel	droplet	eject	jetter	coater	printhead
ink + jet	0.91	0.9	0.86	0.84	0.77	0.76	0.76	0.76	0.75	0.74

	paperless	microfilm	paperlik	microprint	paperi	paper	microfich	papermak
paper + micro	0.81	0.8	0.78	0.77	0.75	0.75	0.75	0.74

	immunotherapi	cancer	inflammasom	immunoinflammatori	immunotherapeut	adenomyosi
vaccin + cancer	0.91	0.91	0.89	0.89	0.89	0.89

Mesures de similarité entre documents

On utilise les vecteurs pour calculer des **« distances technologiques »** entre les brevets (en vue de pouvoir constituer des « clusters technologiques », de mesurer la nouveauté etc.).

Distance utilisée: **cosinus** (angle séparant deux vecteurs).

Evaluer la qualité des plongements

Principe : une série de tests comparant les résultats issus des plongements avec des valeurs de référence issues de jugements humains
⇒ Liens de citation, classification, inventivité

Statistique utilisée: AUC (« Area Under the Curve »)
Statistique standard dans le domaine de la classification
Le score AUC compare le résultat du modèle avec un tirage aléatoire :

- Un score proche de 0.5 signifie que le modèle ne fait pas mieux qu'un tirage au hasard
- Un score proche de 1 (ou de 0) signifie que le modèle discrimine efficacement

Test de citations

Question : le modèle prévoit-il le fait que deux brevets seront reliés par un lien de citation ?

Mise en œuvre : on a tiré deux échantillons de couples de brevets OEB, liés par une citation (échantillon 1) ou non (échantillon 2). L'AUC nous dit la probabilité pour que le modèle discrimine entre les deux types de couples.

Test de citations – résultats

(Un score proche de 1 signale une meilleure qualité)

Tableau 2 : test des citations (AUC)	
Modèle	AUC
doc2vec racinisé EPO-USPTO 12 dimensions	0,959
doc2vec racinisé EPO 12 dimensions	0.971
doc2vec lemmatisé EPO 12 dimensions	0.974
Fasttext EPO-USPTO 40 dimensions	0.970
TF-IDF EPO lemmatisé	0.971
TF-IDF EPO racinisé	0.972

La problématique

Complexités spatiales:

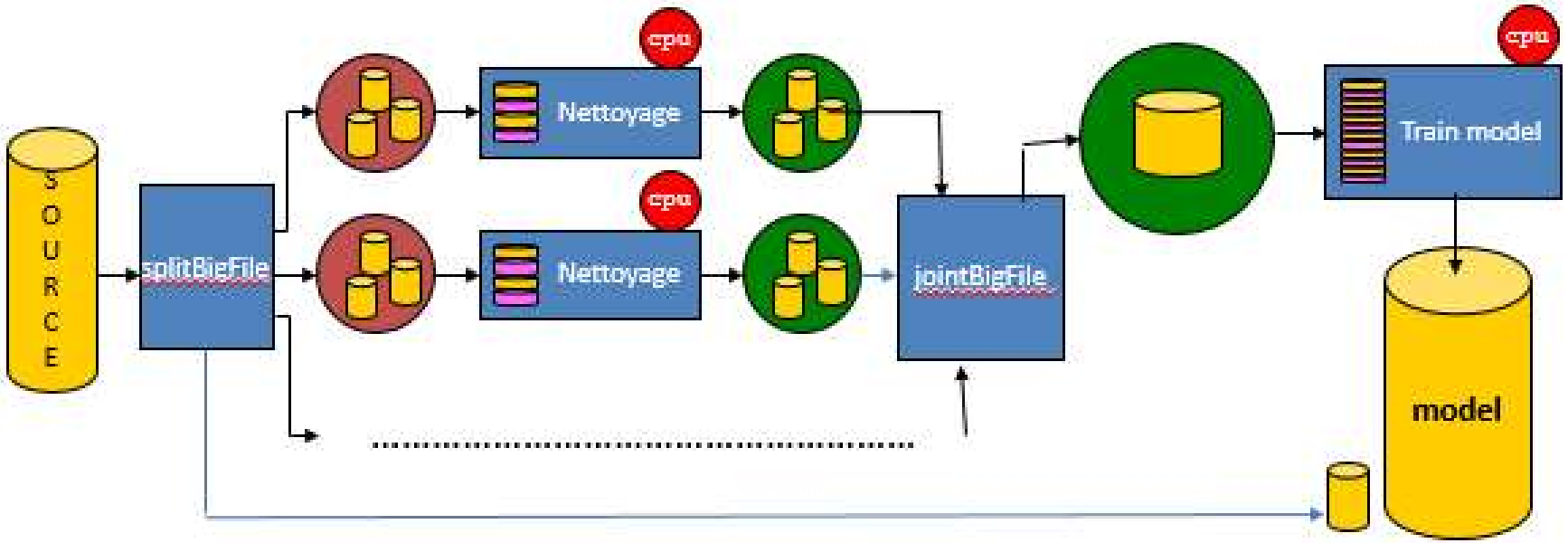
- textes de corpus de brevets: ~12 millions de brevets
- données par Giga octets (textes, embeddings, …)

Complexités temporelles:

- Traitement de textes
- Calculs matricielles
- Tris

Exigence: temps de calcul des indicateurs < 12 heures

Optimisation des traitements – embeddings

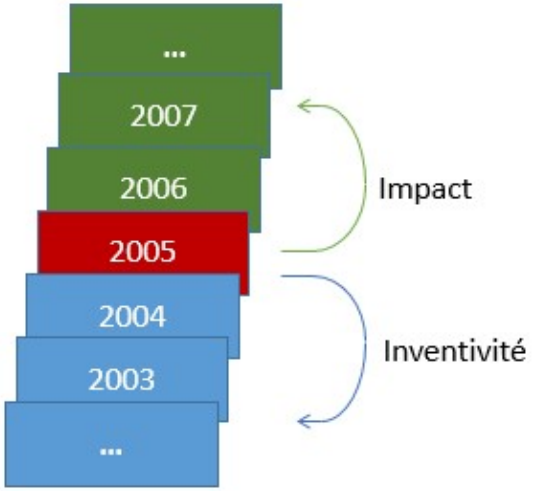


Traitements parallèles de textes : MPI, multiprocessing
Entrainement des modèles doc2vec, … : multithreading

Extraction de l'embedding à partir du modèle

Extraction des embeddings (+ année)

app	annee	0	1	2	3	4
323911681	2009	-0.510529	0.215014	-0.307937	-0.0904332	-0.0608
16285698	2005	0.207282	0.293372	0.101319	-0.168779	-0.1966
16154631	2003	0.047912	0.204101	0.0708023	-0.117342	-0.4140
57638920	2007	-0.240466	0.436515	-0.144219	-0.0524918	0.05677
16351243	2005	0.56006	0.00946262	-0.465982	-0.0669971	0.14479
458653707	2015	0.039313	0.303785	-0.343646	-0.0532105	-0.3875



Calcul des indicateurs à partir des embeddings

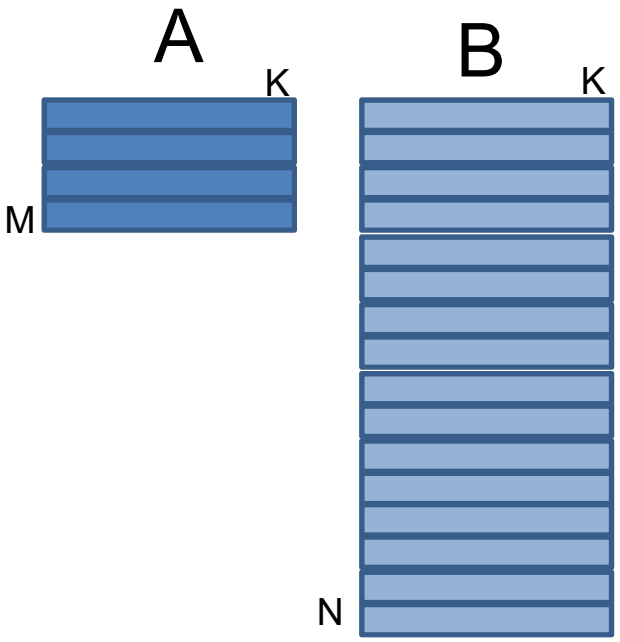
Pour une année donnée:

Matrice A: sélection des embeddings d'une année

Matrice B: sélection des embeddings des années antérieurs (ou postérieurs)

Exploitation des embeddings: La problématique

APPLN_FILING_YEAR	NOMBRE
2010	457787
2009	439427
2008	467601
2007	478684
2006	478861
2005	489173
2004	450201
2003	402945
2002	382158
2001	374525



$$C_{M,N} = A_{M,K} * B^{T}_{N,K}$$

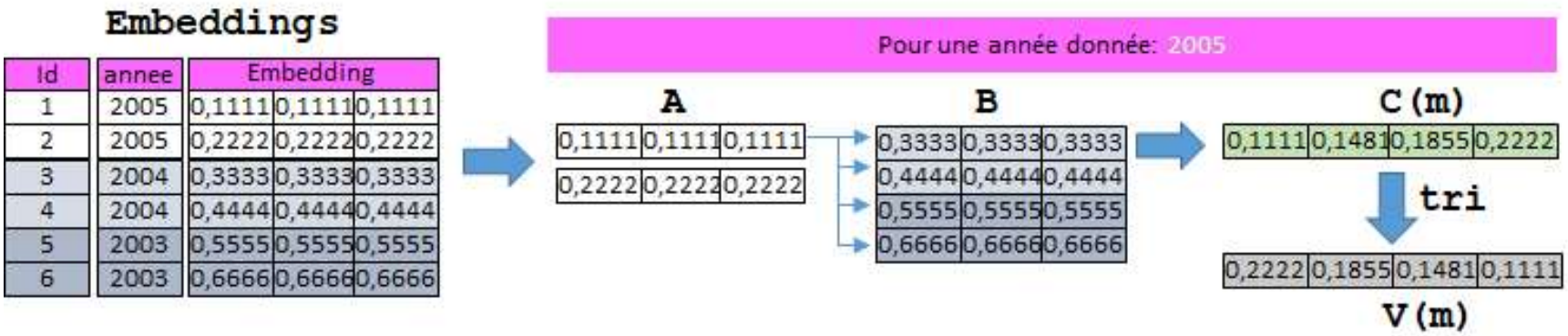
Tri ($C_{M,N}$)
par lignes

Par exemple: M=300.000, K=12, N=1.000.000
$C_{M,N}$ → 7.200.000.000.000 octets (7.2 To)

Optimisation du calcul: 8 solutions évaluées

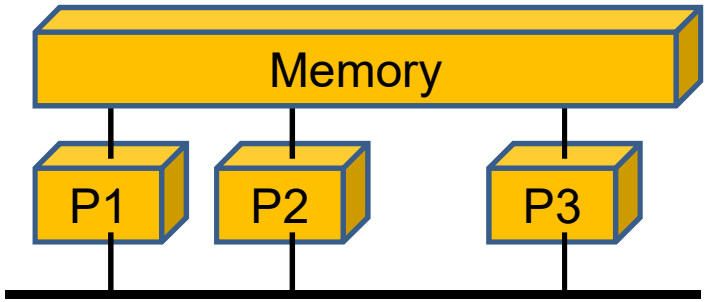
Solution 1 : Python – mono CPU
Solution 2 : Python – multiCPU
Solution 3 : Python – MPI
Solution 4 : CPP – monoCPU
Solution 5 : CPP – MPI
Solution 6 : CPP – monoGPU basique (cuda)
Solution 7 : CPP – monoGPU Thrust/Cublas
Solution 8 : CPP – multiGPU & MPI

Solution 1: Python – monoCPU




```
Pour chaque année de 1990 à 2018:
   Extraction des matrices A et B
   Pour chaque ligne m de A:
      C[m] = B * A^T[m]
      V[m] = tri(C[m], décroissant)
```

Solution 2: Python – multiprocessing

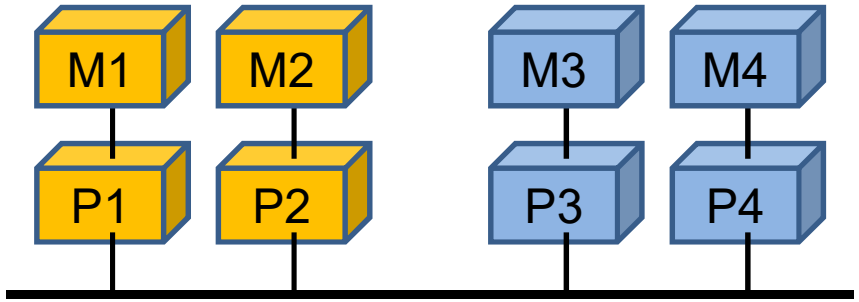


Utilisation du package python: *multiprocessing.Pool*

Le traitement d'une année (de la solution 1) = 1 processus

Nombre de processus = nombre des années à traiter

Solution 3: Python – MPI



Utilisation du package python: *mpi4py*

Distribution des tâches sur plusieurs CPU, une mémoire distribuée, privée pour chaque CPU

Une tâche: traitement d'une année, calée sur un ***rank***

Rank = numéro du CPU parmi les n CPU réservés du cluster

Année = 1990 + rank

Solution 4: CPP – monoCPU

C’est la solution 1 qui est réécrite en C/C++
(une boucle sur les années à traiter, incluant une bouche sur chaque brevet d’une année)

Multiplication matrice x vecteur: basique

Tri de vecteur: quickSort de STL (Standard Template Library)

Solution 5: CPP – MPI

C’est la solution 3 qui est réécrite en C : MPI en C

Une tâche: traitement d’une année, calée sur un ***rank***

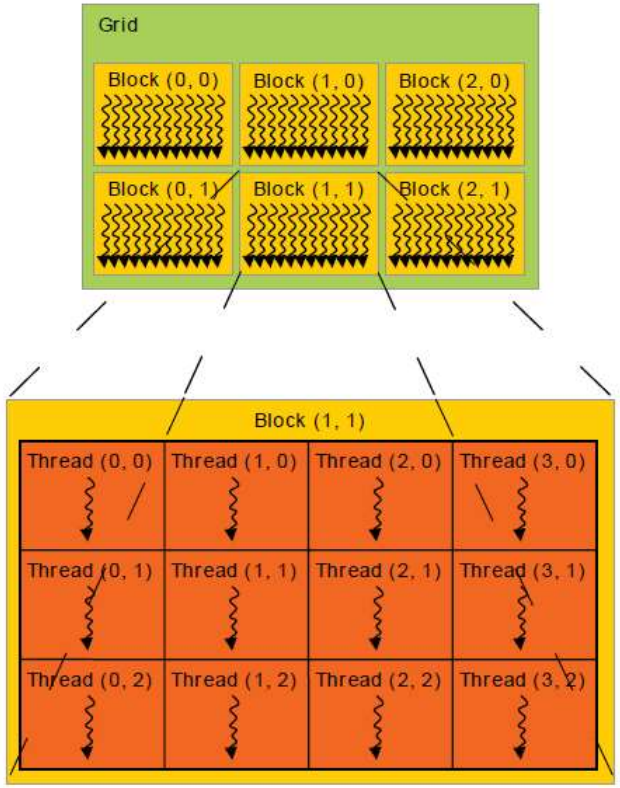
Solution 6: CPP (cuda) – monoGPU (1)

L'idée est d'exploiter les milliers des cœurs du GPU

Un cœur = un thread

Traiter chaque occurrence de la matrice A par un thread:

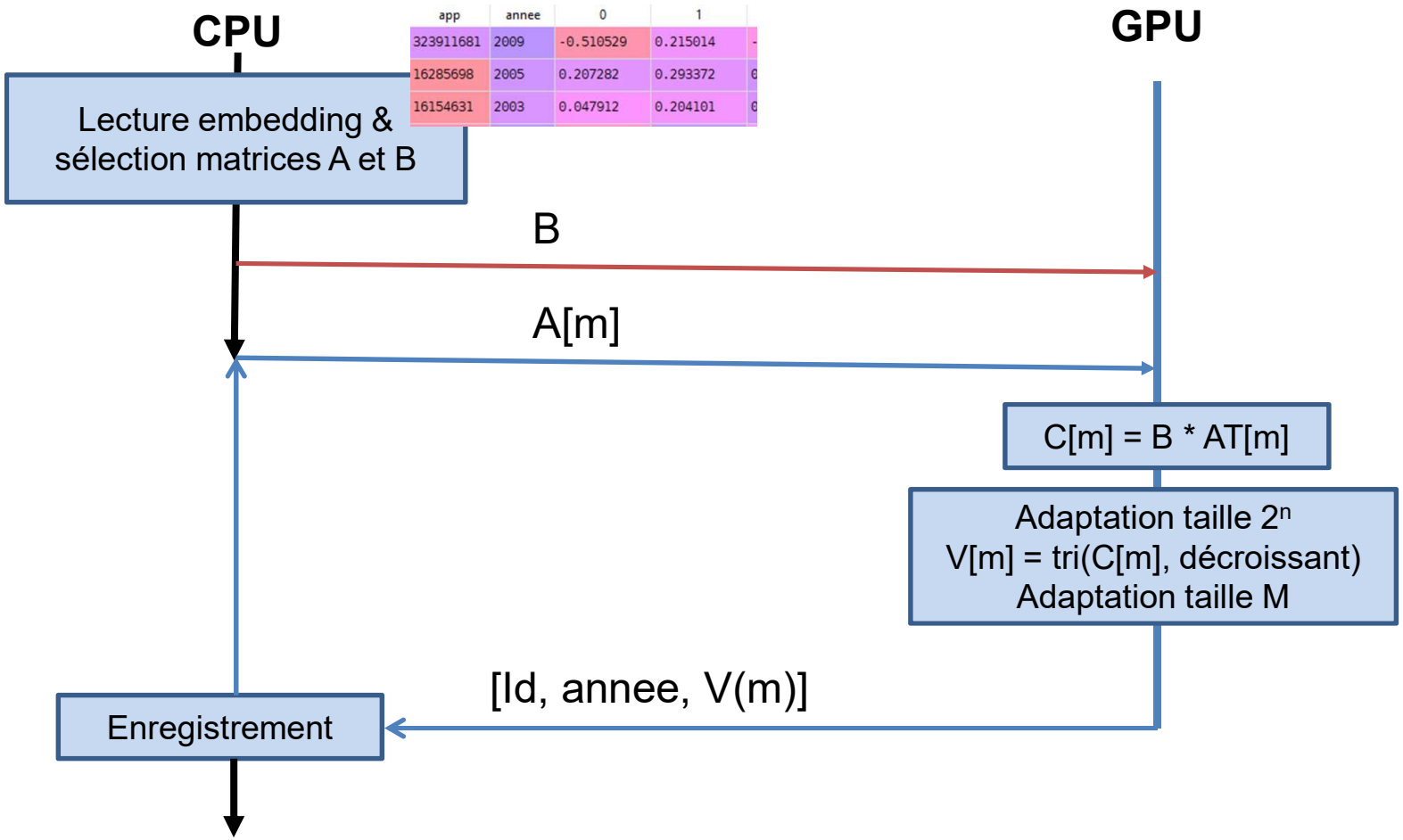
- multiplication de B par une occurrence de A
- tri du vecteur résultant (bitonicSort adapté).



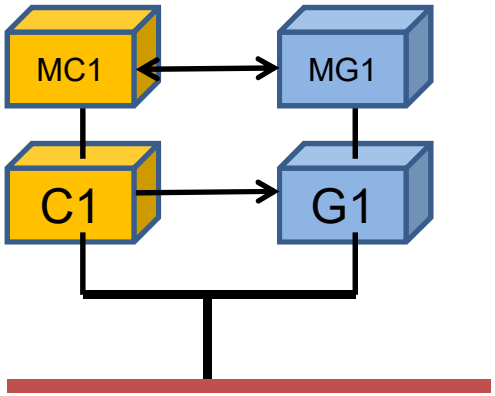

```
int bloc = 32;
int grid = N/bloc + 1;
kernel_computeIndicateurs<<<grid, bloc>>>(A[m], B, C[m]);
```

```
int idx = blockIdx.x * blockDim.x + threadIdx.x;
```

Solution 6: CPP (cuda) – monoGPU (2)



Solution 7: CPP (Thrust & cublas) – monoGPU

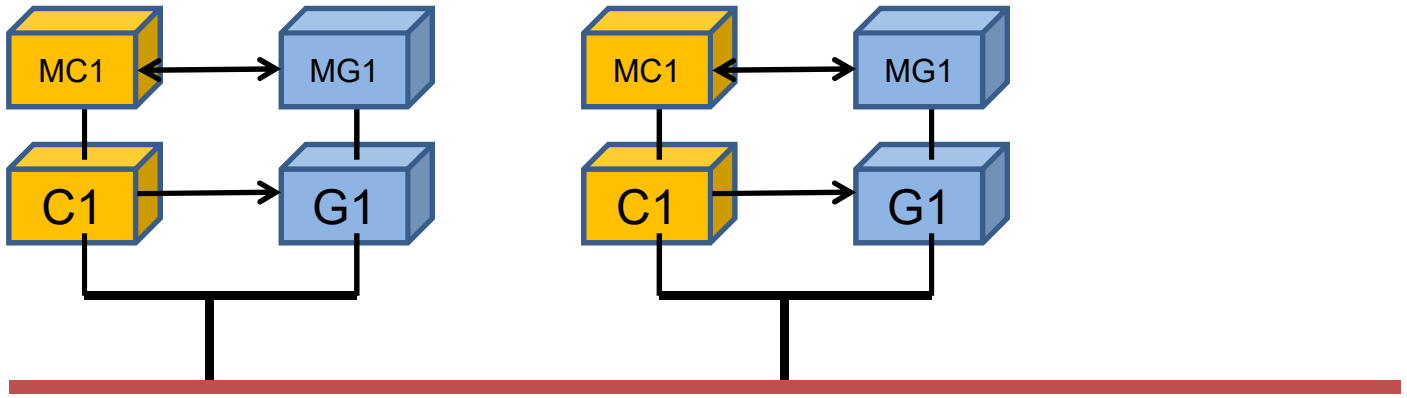


Boucle sur chaque année
Boucle sur la matrice A

- Une tâche: fait le traitement d'une occurrence de A
- Multiplication de matrice x vecteur: cublas
- Tri de vecteur : sort de Thrust (radixSort)

➔ Gestion efficace et transparente de la mémoire

Solution 8: CPP (Thrust & cublas) – multiGPU/MPI



Combinaison MPI et GPU:
Traitement d'une tâche par GPU
Distribution MPI des tâches sur GPU avec ***rank*** = numéro GPU dans le nœud/cluster

Optimisation du calcul: résultat (1)

Solution Echantillon	Python monoCPU	Python multiprocessing	Python MPI4PY	CPP monoCPU	CPP MPI	Cuda monoGPU	Thrust/Cublas monoGPU	Multi-GPU
10K	1.31	1.57	0.44	2.02	0.35	1.66	3.15	1.04
100K	133.20	134.73	23.86	44.39	6.24	20.71	26.50	3.66
500K	3779.51	3604.88	695.57	596.04	101.09	270.73	173.83	25.50
1000K	16421.77 1	10332.88 1.6	2928.24 5.6	2131.17 7.7	369.52 44.5	983.65 16.7	373.84 44.0	51.36 321.0
1500K	>43000	18610.56	6849.17	4568.70	808.79	2321.65	624.13	86.36
2000K		>43000	12544.60	7897.00	1604.75	4297.64	950.46	128.94
2500K			19692.21	12132.91	2476.44	6915.69	1331.52	179.25
~12 millions								

Temps de calcul, en seconde, des différentes solutions pour différents échantillons
[pour 8 années : 2000 à 2007]

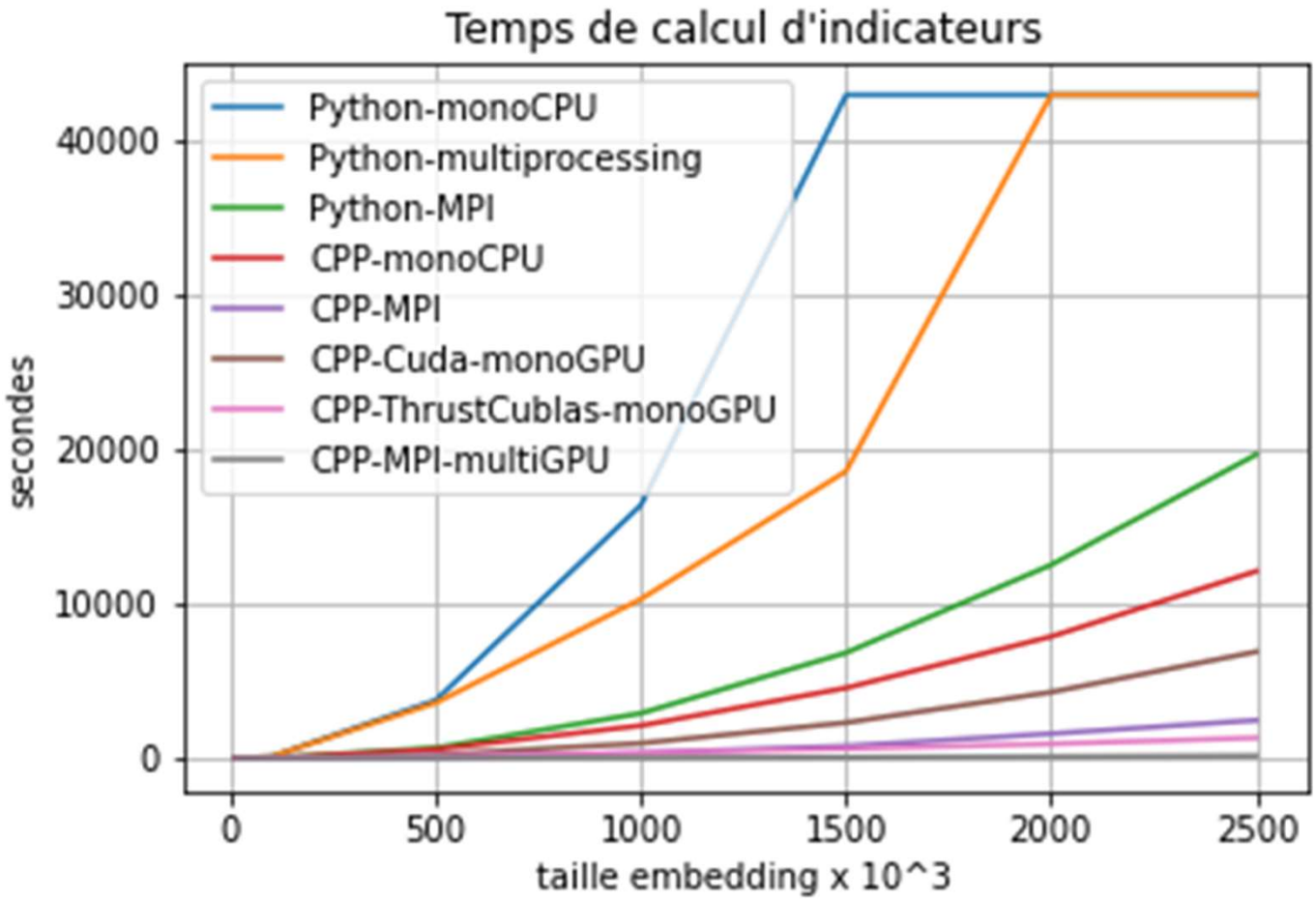
Optimisation du calcul: résultat (2)

Solution 8: GPU / MPI

Embeddings de 8.511.444 occurrences

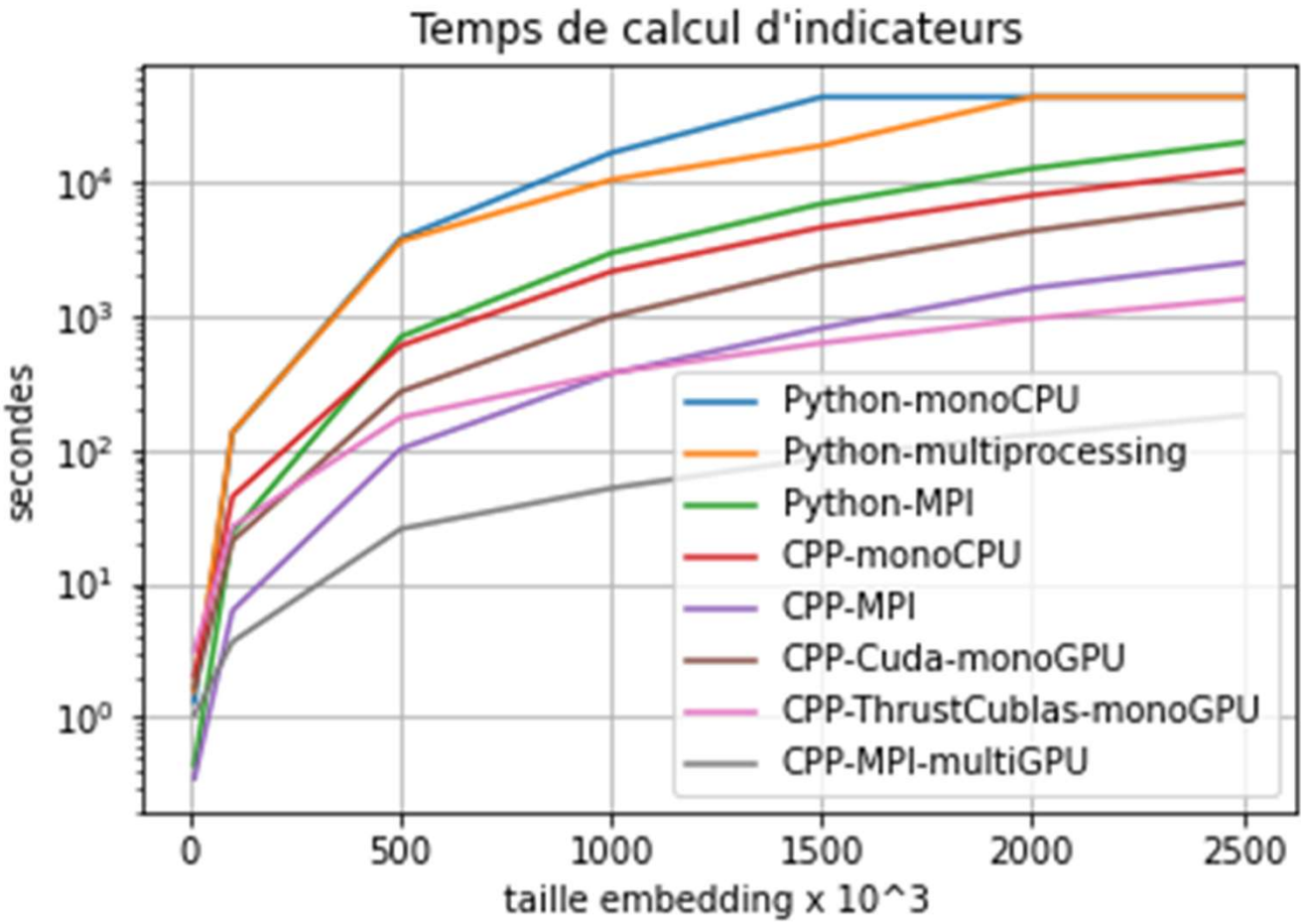
Année	Temps en secondes K=12	Temps en secondes K=48
2004	182.99	1545.92
2005	193.30	1681.04
2006	209.87	1810.40
2007	213.81	1869.77

Optimisation du calcul: résultat (3)



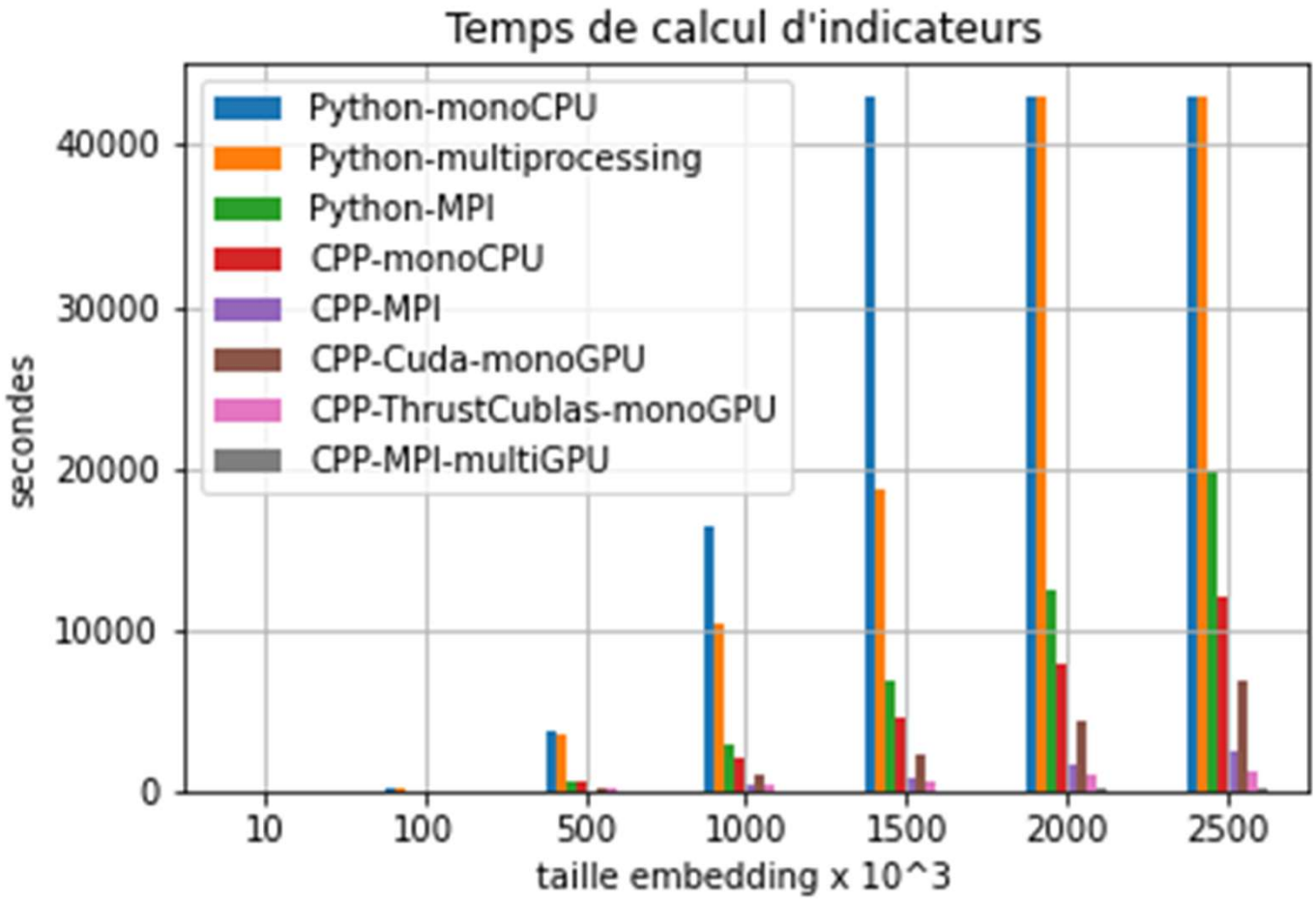
Représentation sur une échelle linéaire

Optimisation du calcul: résultat (4)



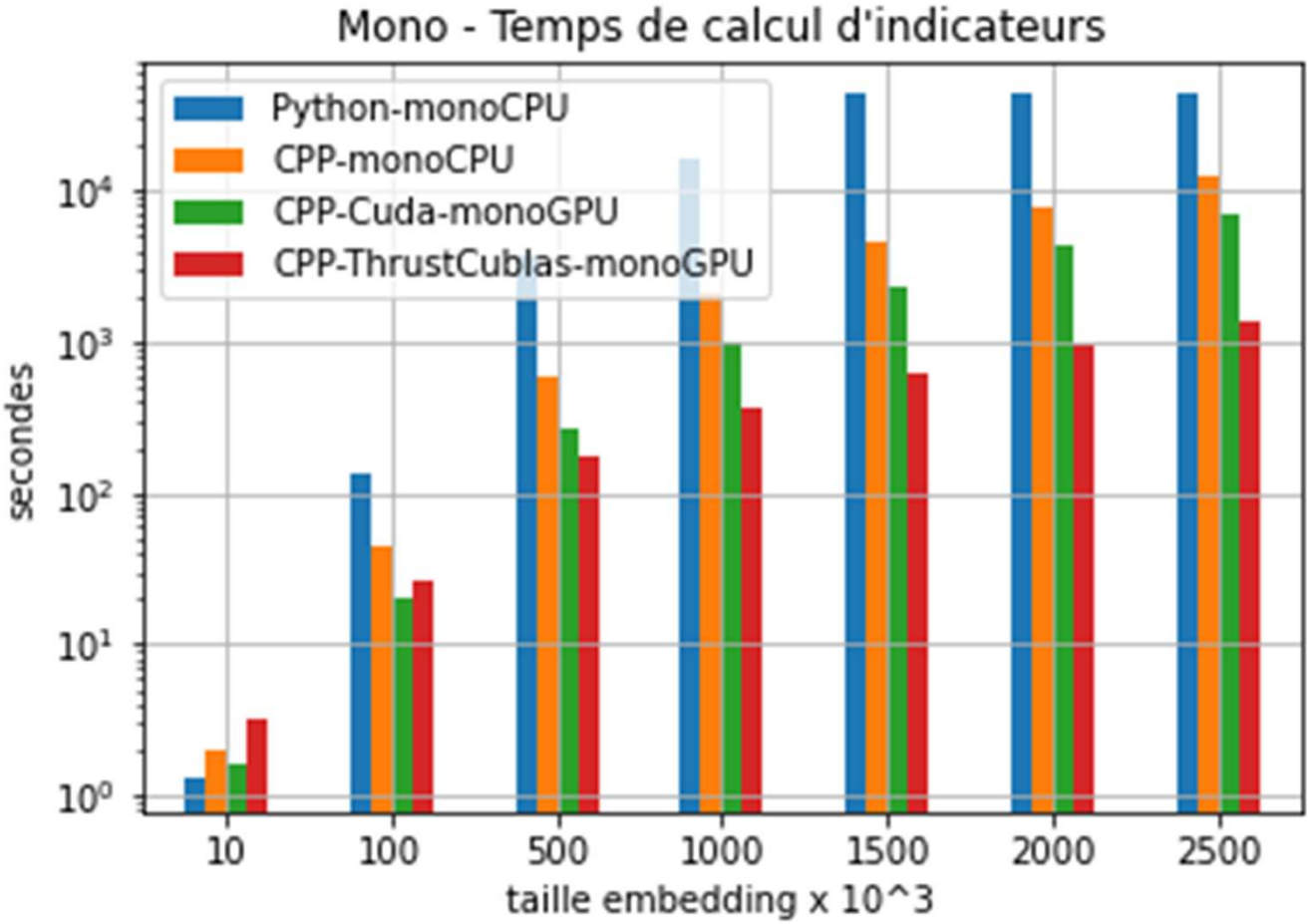
Représentation logarithmique sur l'axe Y

Optimisation du calcul: résultat (5)



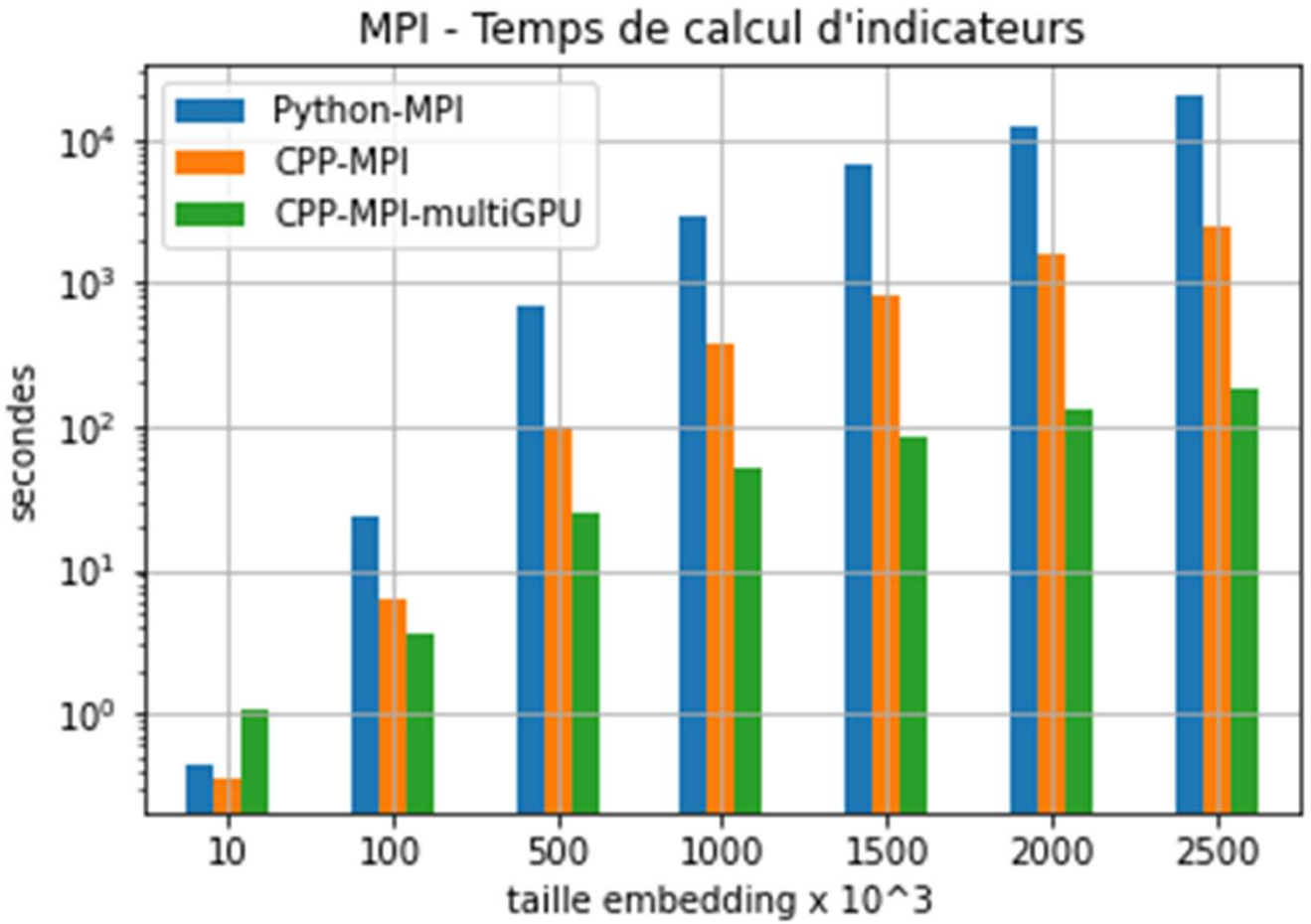
Représentation sur une échelle linéaire

Optimisation du calcul: résultat (6)



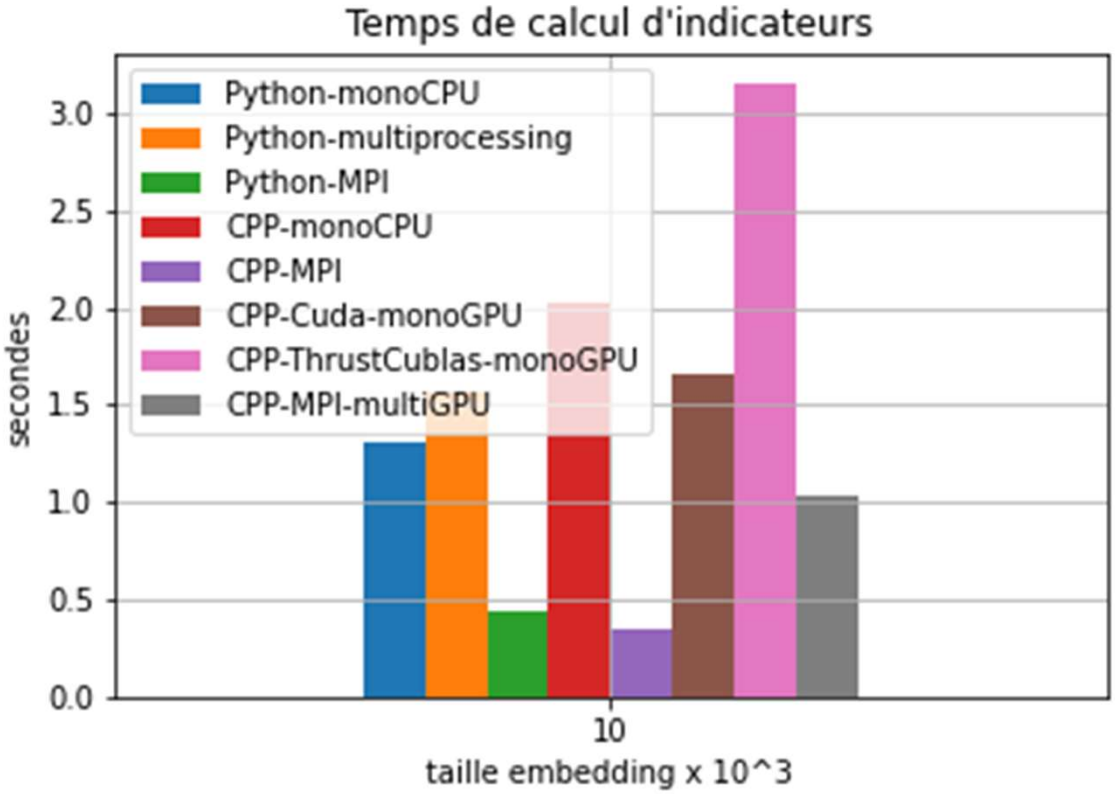
Comparaison entre mono CPU/GPU

Optimisation du calcul: résultat (7)



Comparaison entre CPU/GPU / MPI

Résultats sur petits échantillons



Croisement des courbes pour des solutions Python/CPP/GPU sur les petits échantillons?

→ **Temps de latence (initialisation multiprocessing, MPI, GPU, échange CPU/GPU)**

Conclusion

Nécessité d’utiliser des machines massivement parallèles

La combinaison MPI avec GPU (cublas & Thrust) donne le meilleure résultat

Démarche: conception des algorithmes en Python en monoCPU, puis écriture des solutions finales en C pour une meilleure performance

Perspectives (1)

Tester la solution MPI en combinaison avec GPU sur un grand corpus (embedding de ~12 millions de brevets et pour un vecteur d'embedding K=300 et +)

Optimisations locales:

- Lecture/écriture de fichiers CSV (en une seule passe)
- Traitement par groupes de lignes de la matrice A et non pas ligne par ligne, ni la matrice entière, pour diminuer le nombre de transfert entre CPU et GPU)
- Evaluation d'autres algorithmes de tri adapté au problème

PERSPECTIVES : ARCHITECTURES GPU

MYRIA : PARTITIONS GPU

K80 : 2,91 Tflops DP, 480 GB/s
P100-PCIe-12GB : 4,7 Tflops DP, 550 GB/s
V100-SXM2-32GB :
- 7,8 Tflops DP (*),
- 900 GB/s bande passante mémoire
- Quantité : 20

Calculateur Myria du CRIANN

Myria-2 (2023) : partitions GPU

A100-SXM4-80GB :
- 9,7 Tflops DP,
- 19,5 Tflops DP Tensor cores (cuBLAS …)
- 2039 GB/s bande passante mémoire
- Quantité (cible) : 88

Partitions GPU pour veille technologique
- AMD MI200
- Intel Ponte Vecchio (envisagé)

(*) Double Précision, employée par l'application du HCERES ayant fait l'objet de cette présentation

PERSPECTIVES : ARCHITECTURE CPU

MYRIA : PARTITION CPU

Intel Broadwell (mémoire DDR4)

> 10 000 cœurs

Myria-2 (2023) : partitions CPU

Cibles

- CPU Intel Sapphire Rapids (mémoire DDR5)

> 21 000 cœurs (cible)

- CPU AMD Rome dans serveurs GPU

Gain de puissance effectif CPU (Myria à Myria-2) évalué > 3,3
sur benchmarks d'applications scientifiques de laboratoires normands
(mécanique des fluides, physique des matériaux, dynamique moléculaire)

hceres.com @Hceres_ Hcéres