Parallélisation des algorithmes de recherche de vérité

SOUMAHORO Fanta

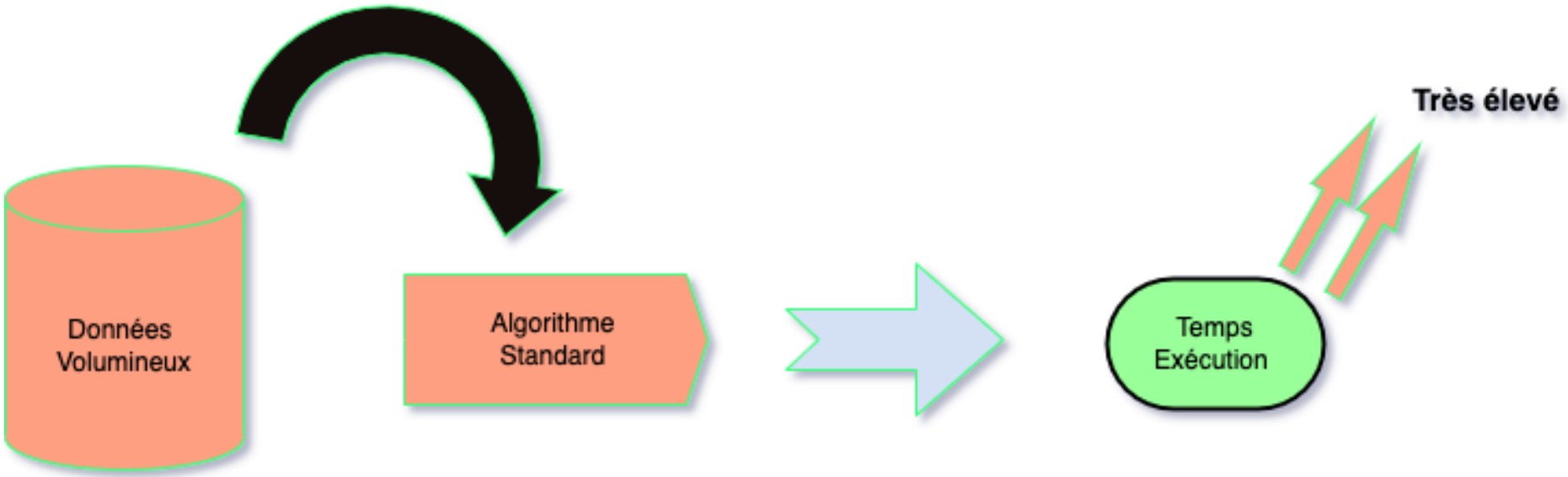
African Institute for Mathematical Sciences, AIMS-Senegal
Sous la supervision de : Dr Mouhamadou Lamine BA
Université Alioune Diop de Bambey (UADB), Sénégal

AIMS | African Institute for Mathematical Sciences SENEGAL

Plan

Contexte & motivation

- Recherche de vérité



- Réduction du temps d'exécution

Définition du problème

Fonction de recherche de vérité F

F prend comme entrée un ensemble de valeurs V fourni par un ensemble de sources S sur un ensemble O d'objets et P de propriétés et retourne deux fonctions $C_v : V \rightarrow [0, 1]$ le score de confiance pour chaque valeur $v \in V$ et $T_s : S \rightarrow [0, 1]$ la fiabilité de chaque source $s \in S$.

- La difficulté du mécanisme de recherche de vérité
- Le caractère hétérogène des données
- La taille des données

Solution : Parallélisation des algorithmes de recherche de vérité.

Exemple

catégorie Géographie	Q1 - Dans quel continent est situé le Brésil ? Q2 - Quelle est la superficie de la Côte d'ivoire ? Q3 - Le Ghana fait-il partie de la CEDEOA ?
catégorie Informatique	Q1- Que signifie RAM ? Q2 - Quel est le langage de progammation le plus utilisé ? Q3 - Qu'affiche ce code python ? *print(6**2)*

Table 1 – Questions concours

postulant	catégorie	Q1	Q2	Q3
postulant 1	Géographie	Afrique	**322,463 km²**	Non
postulant 2	Géographie	**Amérique du Sud**	233 km²	**Oui**
postulant 3	Géographie	**Amérique du Sud**	320,463 km²	Non
postulant 1	Informatique	Random Access Manager	**Python**	12
postulant 2	Informatique	**Random Access Memory**	Java	**36**
postulant 3	Informatique	Random Allow Memory	**Python**	**36**

Table 2 – Réponses concours

Classification des algorithmes de recherche de vérité

Familles	Algorithmes	Auteurs
Méthode probabilistes bayesiens	TruthFinder	Yin et al., 2008
	Découverte de la vérité par corroboration des informations	Galland et al., 2010
	Modèle de vérité latente	Zhao et al., 2012
	Découverte de la vérité par estimation de la vraisemblance maximale	Wang et al., 2012
	Découverte de la vérité avec dépendance des sources	Dong et al., 2009
	Analyse de crédibilité latente	Pasternack et Roth, 2013
Méthodes basées sur l'optimisation	Recherche de la vérité semi-supervisée	Yin et Tan, 2011
Méthodes basées sur partitionnement de données	Recherche de la vérité avec partitionnement des attributs	Ba et al., 2015
	Découverte de la vérité basée sur le partitionnement efficace des données	Osias Noël et Ba, 2021

Table 3 – Classification des algorithmes de recherche de vérité

Proposition d'approche parallèle

Une approche proposée par (Ouyang et al.) en 2016 qui utilise le paradigme MapReduce et a prouvé l'efficacité de la découverte de la vérité sur de grands ensembles de données. Cet algorithme est utilisé dans les applications de crowdsourcing.

- Crowdsourcing est un processus qui consiste à obtenir le contenu, les informations ou les services nécessaires en sollicitant les contributions d'un grand groupe de personnes généralement indéterminées.
- Cet algorithme utilise seulement les données quantitatives.

Algorithmes parallèles vs. algorithmes non parallèles

Critères de parallélisation

Nous considérons un algorithme parallélisable si :

- C_v et T_s sont des sommes
- C_v et T_s sont des produits
- la distribution des données se fait par partitionnement

Algorithme parallèles vs. algorithmes non parallèles

Algorithmes parallélisables

Les algorithmes parallélisables sont :

- MajorityVoting
- TruthFinder
- Recherche de la vérité avec partitionnement des attributs
- Découverte de la vérité basée sur le partitionnement efficace des données
- Recherche de la vérité semi-supervisée
- Depen
- Analyse de crédibilité latente

Algorithmes parallèles vs. algorithmes non parallèles

Algorithmes non parallélisables

Les algorithmes non parallélisables sont :

- Cosine
- 2-estimate
- 3-estimate
- Découverte de la vérité par estimation de la vraisemblance maximale

TurthFinder (Yin, Han et Yu 2008)

- Approche bayesienne avec support mutuel des valeurs similaires. L'algorithme TruthFinder contient les différentes fonctions suivantes :
  - fonction de la fiabilité d'une source (1)
  - fonction de confidence d'une valeur utilisant une fonction logistique (4)
  - fonction de confidence d'une valeur (2)

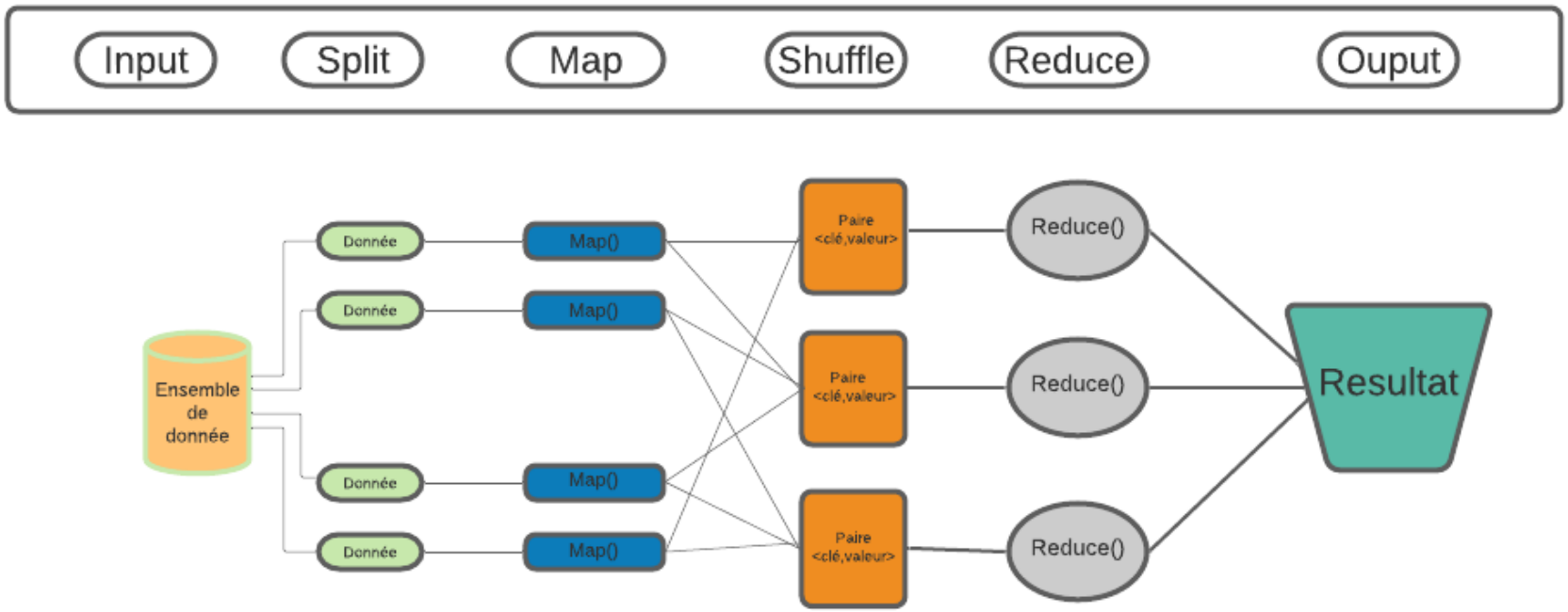
$$T_s = \sum_{v \in V_s} \frac{C_v}{|V_s|} \tag{1}$$

$$\sigma_v = -\sum_{s \in S_v} \ln(1 - T_s) \tag{2}$$

$$\sigma_v^* = \sigma_v + \rho \sum_{v^* \in V_d} \sigma_{v^*}.sim(v, v^*) \tag{3}$$

$$C_v = \frac{1}{1 + e^{-\gamma \sigma_v^*}} \tag{4}$$

Paradigme MapReduce



Version parallèle σ_V

Nous proposons une parallélisation de la fonction de confidence d'une valeur.

Algorithme 1 Fonction de calcul de la confidence d'une valeur :(S,O,A,V,C_v, T_s)

Pré-conditions : $\forall s \in S : T_s \leftarrow 0.8$
Pré-conditions : $\forall v \in V : C_v \leftarrow 0$

1: Écriture de la fonction MAP
2: **Pour tout** $(o, p, v, s, t_s) \in (O, P, V, S, T_S)$ (en parallele) **faire**
3: $\sigma'_v \leftarrow -ln(1 - t_s)$
4: **Fin Pour**
5: Retourne un couple $(< o, p, v >, < \sigma'_v, s, t_s >)$
6: Écriture de la fonction REDUCE
7: **Pour tout** $(< o, p, v >, < \sigma_v, s, t_s >) \in (< O, P, V >, < \sigma_V, S, T_S >)$ (en parallèle) **faire**
8: $\sigma_v \leftarrow \sum_{s \in S_v} \sigma'_v$
9: **Fin Pour**
10: Retourne $(O, P, < (V_i, \sigma_{V_i}), ..., (V_n, \sigma_{V_n}) >, S, T_S)$

Version parallèle C_V

Nous proposons une parallélisation de la fonction de confidence d'une valeur utilisant une fonction logistique.

Algorithme 2 Fonction de calcul de la confidence d'une utilisant une fonction logistique valeur :(S,O,P,V,C_v, T_s,σ_v, ρ, γ)

Pré-conditions : $\gamma \leftarrow 0.5$,$\rho \leftarrow 0.7$
Pré-conditions : $\forall v \in V : c_v \leftarrow \sigma_v$
1: *Écriture de la fonction MAP*
2: **Pour tout** $(o, p, < (v_i, \sigma_{v_i} >, s, t_s) \in (O, P, < (V_i, \sigma_{v_i} >, S, T_S)$ (en parallèle) **faire**
3: $c_v \leftarrow \rho\sigma'_v \times sim(v, v')$
4: $C_v \leftarrow 1/(1 + \exp(-\gamma \times c_v))$
5: **Fin Pour**
6: Retourne un couple $(< o, p >, < v, c_v, s, t_s >)$
7: *Écriture de la fonction REDUCE*
8: **Pour tout** $(< o, p >, < v, c_v, s, t_s >) \in (< O, P >, < V, C_V, S, T_S >)$ (en parallèle) **faire**
9: $C_v \leftarrow \sum_{s \in S_v} c_v$
10: **Fin Pour**
11: retourne (O, P, V, T_S, C_V)

Version parallèle T_s

Nous proposons une parallélisation de la fonction de calcule de fiabilité d'une source.

Algorithme 3 Fiabilité source :(S,O,P,V,C$_v$, T_s)

1: *Écriture de la fonction MAP*
2: **Pour tout** $(s, o, p, v, c_v, t_s) \in (S, O, A, V, C_v, T_s)$ (en parallèle) **faire**
3: $t_s \leftarrow c_v/|V_s|$
4: **Fin Pour**
5: Retourne un couple $(< o, p, v, c_v >, < s, t_s >)$
6: Écriture de la fonction REDUCE
7: **Pour tout** $(< o, p, v, c_v >, < s, t_s >) \in (< O, P, V, C_v >, < S, T_S >)$ (en parallèle) **faire**
8: $T_s \leftarrow \sum_{v \in V_s} t_s$
9: **Fin Pour**
10: Retourne (S, O, A, V, C_v, T_s)

Mesures de performance

Nous effectuons une comparaison entre notre approche et la version séquentielle en utilisant les mesures de performance suivantes :

- L'accuracy
- La précision
- Le recall
- Le f1-score.

Nous prouvons l'efficacité de notre approche avec les mesures de performance suivantes :

- Le temps d'exécution
- La mémoire consommée.

Données de validation

Nous validons notre approche sur deux types de jeux de données

- Données synthétiques
- Données réelles

Données synthétiques

Informations	DS1	DS2	DS3
na	6	10	10
no	1000	1000	2000
ns	10	10	50
td	60.000	100.000	1.000.000

Table 4 – Information concernant les données synthétiques

Performance sur les Données synthétiques

Données	Versions	Recall	Précision	Accuracy	F1-score	Temps(s)	Nbr Iter
DS1	Séquentielle	0.84	0.79	0.89	0.81	2700	2
	Parallèle	0.84	0.79	0.89	0.81	7,8	2

Table 5 – Performance des versions sur le jeu de données synthétique DS1

Données	Versions	Recall	Précision	Accuracy	F1-score	Temps(s)	Nbr Iter
DS2	Séquentielle	0.93	0.88	0.92	0.90	9219	3
	Parallèle	0.93	0.88	0.92	0.90	20,79	3

Table 6 – Performance des versions sur le jeu de données synthétique DS2

Données	Versions	Recall	Précision	Accuracy	F1-score	Temps(s)	Nbr Iter
DS3	Séquentielle	0.96	0.91	0.95	0.93	494049	2
	Parallèle	0.96	0.91	0.95	0.93	252	2

Table 7 – Performance des versions sur le jeu de données synthétique DS3

Temps d'exécution & consommation en mémoire

Données	Versions	Temps(s)
DS1	Séquentielle	2700
	Parallèle	7.8
DS2	Séquentielle	9219
	Parallèle	20.79
DS3	Séquentielle	494049
	Parallèle	252

Table 8 – Temps d'exécution

Données	Versions	Mémoire(Go)
DS1	Séquentielle	0.0502
	Parallèle	0.0009
DS2	Séquentielle	0.0720
	Parallèle	0.0015
DS3	Séquentielle	0.7207
	Parallèle	0.0024

Table 9 – Consommation en mémoire

- Réduction de temps de 99,71% pour le jeux de donnée DS1
- Réduction de temps de 99,77% pour le jeux de données DS2
- Réduction de temps de 99,99% pour le jeux de données DS3

Données réelles

Données	Stocks	Vols
Nombre de sources	55	37
Nombre d'objets	100	100
Nombre d'attributs	15	6
Nombre d'observations	56992	8771

Table 10 – Caractéristiques des jeux de données réelles

Temps d'execution & consommation en mémoire

Données	Versions	Temps(s)
Vols	Séquentielle	48
	Parallèle	2.43
Stocks	Séquentielle	5227
	Parallèle	8.40

Table 11 – Temps d'exécution

Données	Versions	Mémoire(Go)
Vols	Séquentielle	0.0010
	Parallèle	0.0001
Stocks	Séquentielle	0.0101
	Parallèle	0.0008

Table 12 – Consommation en mémoire

- Réduction du temps est de 94,43% pour le jeux de données Vols.
- Réduction du temps est de 99,83% pour les jeux de données Stocks.

Conclusion & perspective

- Les deux versions ont les mêmes performances en terme de precision, accuracy, recall et F1 score
- La version parallèle est efficace en terme de temps d'exécution et de consommation en mémoire.
- Améliorer notre approche en faisant une exécution sur système a plusieurs noeuds pour réduire plus le temps d'exécution et la consommation en mémoire.

Fin

Merci pour votre aimable attention