Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavão

Introduction

Notions

Recursively Enumerated

Monoïdes numériques

Spark

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

Résultats et optimisations Résultats des

# Rapport de Travail d'Etude et de Recherche Utilisation de Spark pour Map-Reduce sur des objets combinatoires

Thomas Foltête Adrien Pavão

M1 Informatique Université de Paris-Sud

Avril 2017



# Sommaire

Rapport de Travail d'Etude et de Recherche

Enumerated

Distributed Implémentations

- 1 Introduction
- Notions préliminaires
  - Recursively Enumerated Set
  - Monoïdes numériques
  - Map-Reduce
- 3 Spark
  - Premiers pas avec Spark
  - Resilient Distributed Datasets
  - Implémentations utilisant des RDDs
- 4 Résultats et optimisations
  - Résultats des approches testées
  - Optimisations des performances
- Conclusion

# Introduction

Rapport de Travail d'Etude et de Recherche

> Thomas Foltête, Adrien Pavã

Introduction

Notions

Recursively Enumerated Set

Monoïdes numériques Map-Reduce

Spark

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des RDDs Framework Spark:

Haut niveau.

Jusqu'à 100 (10) fois plus rapide qu'Hadoop en mémoire (sur disque).

Utilisation classique sur des données stockées.

Combinatoire:

Étudie les collections d'objets, leurs combinaisons et leur dénombrement.

L'informatique est adapté à ce genre de calcul.

### Notions préliminaires **Parallélisation**

Rapport de Travail d'Etude et de Recherche

#### Notions préliminaires

Enumerated

Distributed Implémentations Distribution du calcul pour qu'il soit exécuté simultanément (en parallèle) par plusieurs agents.

- Multi threads (mono machine)
- Multi processeurs (mono machine)
- Multi machines

# Notions préliminaires Recursively Enumerated Set

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavã

#### Introductio

Notions préliminaires

Recursively Enumerated Set

Monoïdes numériques

#### Spark

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

Résultats et optimisation

### Ensemble récursivement énumérable :

- Une racine (élément initial)
- Une fonction successeur

### Notions préliminaires Monoïdes numériques

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavá

Introduction

Notions préliminaire

Recursively Enumerated Set

Monoïdes numériques Map-Reduce

#### Spark

avec Spark
Resilient
Distributed
Datasets
Implémentations
utilisant des

### Numerical semi-group:

- Sous-ensemble de N contenant 0
- Stable par addition
- Complémentaire fini

### Propriétés d'un semi-group :

- Multiplicité
- Genus
- Frobenius
- Conducteur

# Notions préliminaires Exemple d'un monoïde numérique

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavã

#### Introduction

#### Notions préliminaire

Recursively Enumerated

#### Set Monoïdes

numériques Map-Reduc

#### Spark

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

$$S = \{0, 3, 6, 7, 9, 10\} \cup [12, +\infty[= <3, 7>$$

Le sous-ensemble <3, 7> est l'ensemble générateur.

Son ensemble complémentaire est {1, 2, 4, 5, 8, 11}

# Notions préliminaires Map-Reduce

Rapport de Travail d'Etude et de Recherche

Thomas Foltête,

Introductio

Notions préliminaire

Recursively Enumerated Set

Map-Reduce

#### Spar

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des Très utilisé en calcul distribué Se base sur deux fonctions :

- Map : Application d'une fonction à tous les éléments d'une collection.
- Reduce : Réduction de tous les éléments à l'aide d'un opérateur associatif.

## Notions préliminaires Map-Reduce

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavão

Notions préliminaires

Recursively Enumerated

Set Manaïdes

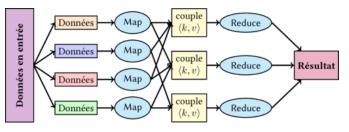
Map-Reduce

#### Sparl

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

Résultats et optimisations Résultats des

Figure – Schéma explicatif du MapReduce



### Premiers pas avec Spark Évaluation et récursivité

Rapport de Travail d'Etude et de Recherche

Enumerated

Premiers pas avec Spark Distributed Implémentations

### Avantage de Spark :

- Évaluation paresseuse : L'évaluation réelle n'est faite que si elle est nécessaire. Résultats non préservés par défaut.
- Aplatissement en mémoire : Fixe les données en mémoire en les évaluant.

# Spark RDD

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavã

Introduction

Notions préliminaire

Recursively Enumerated Set Monoïdes numériques Map-Reduce

Spark
Premiers pas
avec Spark
Resilient
Distributed
Datasets

Implémentations utilisant des RDDs

Résultats et optimisation
Résultats des

**Spark se base sur les RDDs**. Ils sont tolérants à la panne et ont une structure parallèle. Ils permettent à l'utilisateur de :

- Garder les données intermédiaires explicitement en mémoire
- Contrôler leur partitionnement afin d'optimiser l'emplacement des données
- Manipuler les données en utilisant un ensemble important d'opérateurs

#### Notions préliminaire

Recursively Enumerated Set

Monoïdes numériques Map-Reduce

#### Spar

Premiers pa avec Spark Resilient Distributed

Datasets
Implémentations
utilisant des
RDDs

Résultats et optimisation Résultats de Création d'un RDD à partir d'une liste Python :

$$rdd = sc. parallelize([1,2,3,4,5])$$

Il y a deux différents types de méthodes qu'on peux appliquer aux RDDs :

- Les transformations : Union, Map, etc.
- Les actions : First, Reduce, etc.

# Spark

Différentes implémentations utilisant des RDDs

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavá

#### introduction

Notions préliminaires

Recursively Enumerated Set

Monoïdes numériques

Spark

#### Premiers pa avec Spark Resilient Distributed

Implémentatior utilisant des RDDs

optimisation Résultats des **Ensembles des mots d'un alphabet :** Produit cartésien d'un alphabet A avec lui-même.

Implémenté en Python plutôt qu'en Sage.

#### Spark

avec Spark
Resilient
Distributed
Datasets
Implementation

utilisant des RDDs

Résultats et optimisation Résultats des

### Ensemble des arbres binaires à n nœuds :

Cet ensemble se modélise ainsi :

$$\begin{cases}
T_0 = Leaf \\
T_n = \bigcup_i T_i \times Node \times T_{n-i-1}
\end{cases}$$
(1)

On peut ensuite par exemple compter les arbres ou compter les arbres par profondeur. Ce modèle peut également permettre de manipuler des polynômes.

Rapport de Travail d'Etude et de Recherche

Thomas Foltête,

ntraduction

Notions

Recursively

Set
Monoïdes

Map-Reduc

#### $\mathsf{Spark}$

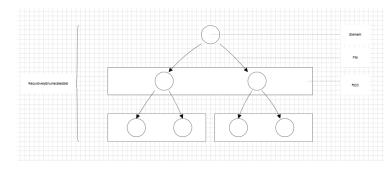
RDDs

avec Spark
Resilient
Distributed
Datasets
Implémentations
utilisant des

Résultats et optimisations Résultats des

# Approche A

# Figure – Schéma explicatif de l'approche A



Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavá

. . .

Notions

préliminaires

Enumerated Set

numériques Map-Reduc

#### $\mathsf{Spark}$

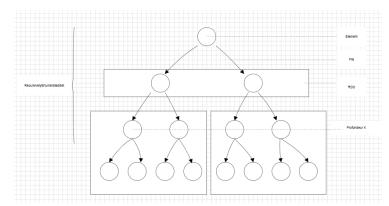
RDDs

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

Résultats et optimisation Résultats des

# Approche B

Figure – Schéma explicatif de l'approche B



Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pay

### Notions

préliminaires

Enumerated Set

numériques Map-Reduc

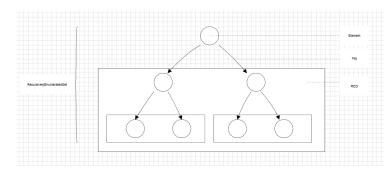
#### $\mathsf{Spark}$

avec Spark
Resilient
Distributed
Datasets
Implémentations
utilisant des

RDDs Résultats et optimisations

# Approche C

# Figure – Schéma explicatif de l'approche C



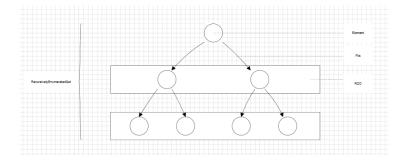
Rapport de Travail d'Etude et de Recherche

RDDs

Implémentation s utilisant des

# Approche D

# Figure - Schéma explicatif de l'approche D



### Résultats et optimisations Critères d'évaluation

Rapport de Travail d'Etude et de Recherche

Enumerated

Distributed Implémentations

Résultats et optimisations

### On observe:

- Le temps de calcul : On souhaite le minimiser
- L'utilisation mémoire : Savoir si tout est développé en mémoire
- La distribution du calcul et le partitionnement des RDDs

### Résultats et optimisations Protocole de test

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavâ

Introduction

#### Notions préliminaire

Recursively Enumerated Set

Monoïdes numériques Map-Reduce

#### Sparl

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

Résultats et optimisations

- Dénombrement des mots de taille n sur l'alphabet A = {a, b, c, d}
- Machine : 2 processeurs à 2.40GHz et 8 Go de RAM
- Count avec un Map-Reduce
- Mesure du temps dans le code Python avec time.time()
- Observation des processus et de la mémoire avec top
- Nombre de partitions du RDD en temps réel avec la méthode getNumPartitions()

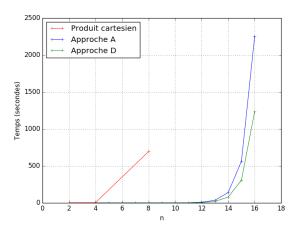
# Résultats et optimisations Résultats des approches testées

Rapport de Travail d'Etude et de Recherche

Distributed **Implémentations** 

Résultats des approches

Figure – Temps d'exécution en fonction de n



## Résultats et optimisations Analyse des premiers résultats

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavê

Introduc

Notions

Recursively Enumerated Set

Monoïdes numériques Map-Reduce

#### Sparl

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

### Conjecture pour ces résultats décevants :

- Mauvaise parallélisation due à notre implémentation Python, ou aux Strings.
- Certains RDDs sont réévalués plusieurs fois.
- Création récursive de petits RDDs qui prend beaucoup de temps.

## Résultats et optimisations Analyse des premiers résultats

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavâ

Introduction

Notions

Recursively Enumerated Set

Monoïdes numériques Map-Reduce

#### Sparl

Spark
Premiers pas
avec Spark
Resilient
Distributed
Datasets
Implémentations
utilisant des

Résultats des

approches

### Idées de solution :

- Faire du profiling.
- Aplatir le RDD en mémoire à un certain niveau.
   L'aplatissement semble permettre un grand grain de performance.
- Faire un algorithme hybride.

# Optimisations de performances Aplatissement naïf

Rapport de Travail d'Etude et de Recherche

Distributed Implémentations On essai de collecter le RDD dans sa totalité. Saturation de la mémoire pour n > 12.

$$n = 12$$

count = 16777216

Temps d'execution: 4.88712310791 secondes.

# Optimisations des performances

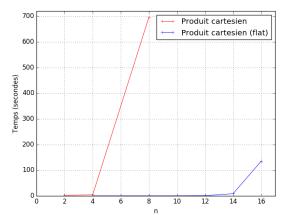
Produit cartésien avec aplatissement mémoire

Rapport de Travail d'Etude et de Recherche

Enumerated

Implémentations

Figure – Schéma comparatif du produit cartésien avec son homologue avec aplatissement mémoire



# Optimisations des performances

Produit cartésien avec aplatissement mémoire

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavâ

Introduction

Notions

Recursively Enumerated Set

Monoïdes numériques Map-Reduc

### Spark

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des RDDs

# Commande top:

PID USER	[]	CPU	MEM	TIME+ CC	DMMAND
29700 ubuntu	[]	99.7	0.3	1:07.20	python
29701 ubuntu	[]	99.7	0.3	1:06.79	python
29705 ubuntu	[]	99.7	0.3	1:06.62	python
29709 ubuntu	[]	99.7	0.3	1:06.78	python

Il s'agit pour le moment du meilleur temps que nous ayons obtenu :

n = 16

count = 4294967296

Temps d'execution: 129.296506166 secondes.

# Optimisations des performances Approche D améliorée

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavão

Notions

Recursively Enumerated

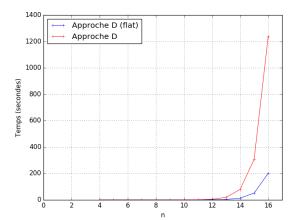
Set Manaides

numériques Map-Reduc

#### $\mathsf{Spark}$

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

Résultats et optimisations Résultats des Figure – Schéma comparatif de l'approche D et de sa version améliorée avec aplatissement en briques de taille 1024



# Optimisations de performances Aplatissement selon la profondeur

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavão

Notions

Recursively

Enumerated Set

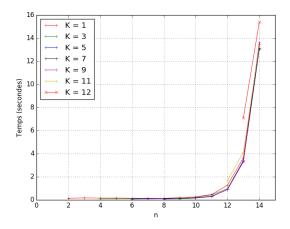
numériques Map-Reduc

#### Spark

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des

optimisations Résultats des

Figure – Temps d'exécution en fonction de n pour différents K



### Résultats et optimisations Vol de tâches

Rapport de Travail d'Etude et de Recherche

Thomas Foltête, Adrien Pavi

Introduction

### Notions

préliminaires

Recursively Enumerated

Set Monoïdes

numériques Map-Reduc

#### Spark

Premiers pas avec Spark Resilient Distributed Datasets Implémentations utilisant des Un arbre déséquilibré en largeur :

```
rdd = sc.parallelize(range(1,5))
for i in range(n):
    rdd = rdd.flatMap( \\
    lambda x : range(1,4*x+1))
```

Les processus s'arrêtent chacun leur tour.

### Résultats et optimisations Utilisation mémoire

Rapport de Travail d'Etude et de Recherche

**Implémentations** 

# Exemple d'utilisation mémoire pendant un calcul avec Spark:

PID USER	[]	CPU	MEM	TIME+	COMMAND
8214 ubuntu	[]	71.1	0.3	0:02.14	python
18122 ubuntu	[]	43.2	3.9	0:12.74	java
18207 ubuntu	[]	12.6	0.3	0:00.38	python
18210 ubuntu	[]	4.3	0.3	0:00.13	python
18211 ubuntu	[]	4.3	0.3	0:00.13	python
18217 ubuntu	[]	3.3	0.3	0:00.10	python

# Conclusion

Rapport de Travail d'Etude et de Recherche

# Enumerated

Implémentations

Possibilité de calcul combinatoire

- Un peu lent mais plus rapide en Scala
- Peut-être pas de vol de tâches prévu par Spark
- Les structures de données influent grandement les performances et la distribution du calcul

Notre étude de Spark ne répond pas forcément à toutes les questions qui l'entourent, cependant ce projet nous aura été très pédagogique.