


Projet 4:

Analyse des données en batch

Par: RASAMBATRA Freonel Carolio



Sommaire:

1. Introduction
 - a. Présentation du sujet
 - b. Présentation des données d'entrées
 - c. Enjeux
 2. Description de stockage:
 - a. Structure de stockage
 - b. Sécurisation
 3. Sérialisation avec avro:
 - a. Sérialisation de history contribution
 - b. Sérialisation de pagelinks
 4. Requêtes Spark SQL
 5. Résultats
 6. Architecture de notre système
 7. Conclusion
- 

1. a) Introduction: présentation du sujet

Déterminer le plus grand contributeur à un sujet donné à partir d'une base de donnée:

- Origine de notre données est Wikipedia
 - Historique de contribution des articles
 - les liens entre les articles (pages)
- Créer notre propre data lake : Dan & Briggs
 - Stratégie de structure des stockage
 - Sécurisation des données
 - Distribution de stockage
==>HDFS
- Sérialiser les données dans notre data lake: facilite leur lecture
- Réaliser une requête pour trouver le candidat idéal



1. b) Introduction: présentation des données d'entrées

Fichier 1: "*frwiki-20200201-stub-meta-history.xml*" :(75,9 Go) historique des contributions aux articles ==> nb de contribution de chaque contributeurs

```
<page>
  <title>Station de Vierville-sur-Mer</title>
  <ns>0</ns>
  <id>13100014</id>
  <redirect title="Chemins de fer du Calvados" />
  <revision>
    <id>166983967</id>
    <timestamp>2020-02-02T01:14:20Z</timestamp>
    <contributor>
      <username>Ldgdps</username>
      <id>2613405</id>
    </contributor>
    <comment>Nouvelle page : a</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text bytes="1" id="167546520" />
    <sha1>frkhg3ewxov0hlg2eh87fri7zlg12ns</sha1>
  </revision>
  <revision>
    <id>166984020</id>
    <parentid>166983967</parentid>
    <timestamp>2020-02-02T01:16:51Z</timestamp>
    <contributor>
      <username>Ldgdps</username>
      <id>2613405</id>
    </contributor>
    <comment>Page redirigée vers [[Chemins de fer du Calvados#Arrêts et gares]]</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text bytes="60" id="167546573" />
    <sha1>33oqqocjbpdn6gc5uh9xvnhlo6rytml</sha1>
  </revision>
</page>
```

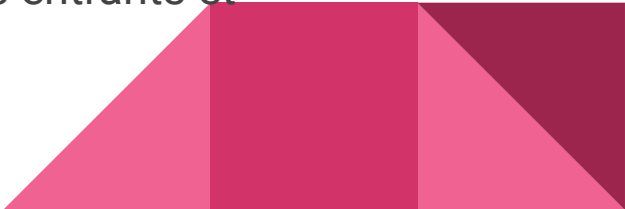
1. b) Introduction: présentation des données d'entrées

Fichier 2: “*frwiki-20200201-pagelinks.sql*” :(11,9 Go) base de données MySql qui contient les lien entre tous les articles

```
DROP TABLE IF EXISTS `pagelinks`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `pagelinks` (
  `pl_from` int(8) unsigned NOT NULL DEFAULT '0',
  `pl_namespace` int(11) NOT NULL DEFAULT '0',
  `pl_title` varbinary(255) NOT NULL DEFAULT '',
  `pl_from_namespace` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`pl_from`, `pl_namespace`, `pl_title`),
  KEY `pl_namespace` (`pl_namespace`, `pl_title`, `pl_from`),
  KEY `pl_backlinks_namespace` (`pl_from_namespace`, `pl_namespace`, `pl_title`, `pl_from`)
) ENGINE=InnoDB DEFAULT CHARSET=binary ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=8;

/*!40000 ALTER TABLE `pagelinks` DISABLE KEYS */;
INSERT INTO `pagelinks` VALUES (177374,0,'!',0),(315352,0,'!',0),(11900520,0,'(1)_Cérès',0);
INSERT INTO `pagelinks` VALUES (11900522,0,'(1)_Cérès',0),(11900524,0,'(1)_Cérès',0),(11900526,0,'(1)_Cérès',0);
INSERT INTO `pagelinks` VALUES (10796149,0,'(10)_Hygie',0),(10796150,0,'(10)_Hygie',0),(10796151,0,'(10)_Hygie',0);
/*!40000 ALTER TABLE `pagelinks` ENABLE KEYS */;
```

1. c) Introduction: enjeux

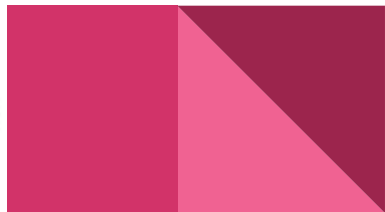
1. Sécuriser les données: HDFS
 - a. Accident de manipulation des données : revenir à un état donnée
 - b. Eviter à tout prix de perdre les données: verrouillage d'accès en écriture
 - c. No single point of failure: distribuer le stockage des données en les dupliquant sur plusieurs machines pour ne pas dépendre que d'une seule machine
 2. Faciliter la lecture: Avro
 - a. Sérialisation des données une seule fois
 - b. Schéma donne un aperçu des données et préparer les requêtes
 3. Prendre en compte toutes les articles relatifs au sujet: graphFrame
 - a. Considérer l'importance des articles grâce à ses liens entrants et sortants
 - b. utilisation structure de graphe orientée
- 

2.a) Description de stockage: Structure de notre data lake

- **/data/** : Contient tous nos données
- **/data.snapshot/s20200720-163848.488/*** : contient le snapshot de tous nos données
- **/data/frwiki/** : contient tout ce qui provient de Wikipedia
- **/data/frwiki/raw/** : contient tous nos données brutes
- **/data/frwiki/raw/frwiki-20200201-pagelinks.sql** : fichier brute contenant les liens entre les pages (11.04 GB)
- **/data/frwiki/raw/frwiki-20200201-stub-meta-history.xml** : fichier brute contenant les informations et historique des pages (70.72 GB)
- **/data/frwiki/frwiki-20200201/** : contient tout ce qui provient des données brutes du 1er février 2020
- **/data/frwiki/frwiki-20200201/master/** : contient tous les jeux de données provenant des données brutes wikipédia du 1er février 2020
- **/data/frwiki/frwiki-20200201/master/pagelink.avsc** : schéma avro utilisé pour sérialiser les liens des pages
- **/data/frwiki/frwiki-20200201/master/pageshistory.avsc** : schéma avro utilisé pour sérialiser l'historique et les informations concernant les pages
- **/data/frwiki/frwiki-20200201/master/full/** : contient tous les données sérialisé (complet)
- **/data/frwiki/frwiki-20200201/master/full/frwiki-20200201-pagelinks-*.avro**: pages links sérialisés
- **/data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-*.avro** : history sérialisé
- **/data/frwiki/frwiki-20200201/master/test/** : contient des échantillons de jeux de données sérialisés ==> pour faire des tests de traitement ou d'observer des de nos données

2.a) Description de stockage: Structure de notre data lake

```
/data/  
  .snapshot/*  
  frwiki/  
    raw/  
      frwiki-20200201-stub-meta-history.xml  
      frwiki-20200201-pagelinks.sql  
  
    frwiki-20200201/  
      master/  
        pageshistory.avsc  
        pagelink.avsc  
        full/  
          frwiki-20200201-pagelinks-*.avro  
          frwiki-20200201-stub-meta-history-*.avro  
        test/  
          *.avro
```



2.b) Description de stockage: Sécurisation des données

1. No single point of failure:
 - a. 3 datanodes sur lesquels les données sont distribuées en blocs
 - b. Namenode secondaire: préserver les données du namenode en faisant des checkpoints régulièrement (toutes les heures)
2. Snapshot: permet de sauvegarder l'état d'un répertoire à un instant t donné
 - i. Rendre le répertoire `"/data"` snapshotable: `hdfs dfsadmin -allowSnapshot /data`
 - ii. Créer des snapshot régulières au rep `"/data"`: `hdfs dfs -createSnapshot /data`
 - iii. Restaurer: `hdfs dfs -cp -f /data/.snapshot/s20200720-163848.488/* /data/`
3. Interdire l'accès en écriture sur les deux répertoire contenant les données brutes et données sérialisés (master dataset):
 - i. bash: `hdfs dfs -chmod -R ugo-w /data/frwiki/raw /data/frwiki/frwiki-20200201/master/full`

/data/frwiki/raw								Go!		
Show	25	entries		Search:						
<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name		
<input type="checkbox"/>	-r--r--r--	maiky	supergroup	11.04 GB	Jul 05 14:11	3	128 MB	frwiki-20200201-pagelinks.sql		
<input type="checkbox"/>	-r--r--r--	maiky	supergroup	70.72 GB	Jul 05 13:56	3	128 MB	frwiki-20200201-stub-meta-history.xml		

2.b) Description de stockage: Sécurisation des données

1. Snapshot:

[Hadoop](#)[Overview](#)[Datanodes](#)[Datanode Volume Failures](#)[Snapshot](#)[Startup Progress](#)[Utilities ▾](#)

Snapshot Summary

Snapshottable directories: 1

Path	Snapshot Number	Snapshot Quota	Modification Time	Permission	Owner	Group
/data	2	65536	Mon Jul 20 16:38:48 +0200 2020	rwxr-xr-x	maiky	supergroup

Snapshotted directories: 2

Snapshot ID	Snapshot Directory	Modification Time
s20200720-163848.488	/data/.snapshot/s20200720-163848.488	Mon Jul 20 16:38:48 +0200 2020
snapshot1	/data/.snapshot/snapshot1	Mon Jul 20 16:28:21 +0200 2020

3.a) Sérialisation avec avro: History de contribution

```
1  {
2    "namespace": "danbriggs.frwiki",
3    "type": "record",
4    "name": "History",
5    "fields": [
6      {"name": "p_title", "type": "string"},
7      {"name": "p_namespace", "type": "long"},
8      {"name": "p_id", "type": "long"},
9      {
10       "name": "p_revisions",
11       "type": {
12         "type": "array",
13         "items": {
14           "name": "revision",
15           "type": "record",
16           "fields": [
17             {"name": "r_id", "type": "long"},
18             {"name": "r_parent_id", "type": "long", "default": -911},
19             {"name": "r_timestamp", "type": "long"},
20             {
21               "name": "r_contributor",
22               "type": {
23                 "name": "Contribution",
24                 "type": "record",
25                 "fields": [
26                   {"name": "r_username", "type": "string", "default": "No name"},
27                   {"name": "r_contributor_id", "type": "long", "default": -911},
28                   {"name": "r_contributor_ip", "type": ["null", "string"]}
29                 ]
30               }
31             },
32             {"name": "r_minor", "type": ["null", "string"]},
33             {"name": "r_comment", "type": ["null", "string"]},
34             {"name": "r_model", "type": ["null", "string"]},
35             {"name": "r_format", "type": ["null", "string"]},
36             {
37               "name": "r_text",
38               "type": {
39                 "name": "Text",
40                 "type": "record",
41                 "fields": [
42                   {"name": "r_text_id", "type": "long"},
43                   {"name": "r_text_bytes", "type": "long"}
44                 ]
45               }
46             },
47             {"name": "r_sha1", "type": ["null", "string"]}
48           ]
49         }
50       }
51     ]
52   }
53 }
```

3.a) Sérialisation avec avro: history (output séparé python) ~8h

```
40 def serialize(xml_path, avro_path, hdfs_client, schema):
41     with hdfs_client.read(xml_path) as xml_file:
42
43         pages = []
44         MAX_PAGES_LENGTH = 750 # Constante à mettre à jour selon la capacité de la machine
45         numero_output = 1
46
47         # On va appliquer la lecture avec namespace du XML
48         xmlns = 'http://www.mediawiki.org/xml/export-0.10/'
49         context = ET.iterparse(xml_file, events=("start", "end"))
50         # turn it into an iterator
51         context = iter(context)
52         event, root = next(context)
53         for event, elem in context:
54             if event == 'end' and elem.tag == ET.QName(xmlns, 'page'):
55                 #p=extract_infosPages(elem, xmlns)
56                 pages.append(extract_infosPages(elem, xmlns))
57
58                 if len(pages) > MAX_PAGES_LENGTH:
59                     root.clear()
60                     avro_file = avro_path + '-' + str(numero_output).zfill(5) + '.avro'
61                     sauvegarder_liste(hdfs_client, avro_file,schema, pages)
62                     numero_output += 1
63                     del pages
64                     pages = []
65
66
67     print('Dernier serialization')
68     # On sauvegarde ce qui reste dans page et qui n'a été écrite dans HDFS
69     avro_file = avro_path + '-' + str(numero_output).zfill(5) + '.avro'
70     sauvegarder_liste(hdfs_client, avro_file,schema, pages)
71     print('Fin de la sérialisation, au revoir !!!')
```

3.a) Sérialisation avec avro: history (output séparé python) ~8h

```
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14052.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14053.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14054.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14055.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14056.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14057.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14058.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14059.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14060.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14061.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14062.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14063.avro
Dernier serialization
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-14064.avro
Fin de la sérialisation, au revoir !!!

real    510m18.712s
user    435m8.903s
sys     7m25.196s
```


3.a) Sérialisation avec avro: history (output séparé python) ~8h

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Browse Directory

/data/frwiki/frwiki-20200201/master/full

Go!



Show 25 entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	27.35 MB	Jul 16 03:06	3	128 MB	frwiki-20200201-stub-meta-history-00019.avro	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	24.15 MB	Jul 16 03:07	3	128 MB	frwiki-20200201-stub-meta-history-00020.avro	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	20.89 MB	Jul 16 03:07	3	128 MB	frwiki-20200201-stub-meta-history-00021.avro	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	19.84 MB	Jul 16 03:08	3	128 MB	frwiki-20200201-stub-meta-history-00022.avro	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	24.36 MB	Jul 16 03:08	3	128 MB	frwiki-20200201-stub-meta-history-00023.avro	

3.b) Sérialisation avec avro: pagelinks (output unifié)

```
1 [
2   "namespace": "danbriggs.pagelinks",
3   "type": "record",
4   "name": "pagelink",
5   "fields":
6     [
7       {"name": "pl_from", "type": "long"},
8       {"name": "pl_namespace", "type": "long"},
9       {"name": "pl_title", "type": "string"},
10      {"name": "pl_from_namespace", "type": "long"}
11     ]
12 ]
13
14 ]
```

3.b) S rialisation avec avro: pagelinks (output s par  python) ~37h

```
40 def serialize(sql_path, avro_path, hdfs_client, schema):
41     with hdfs_client.read(sql_path) as sql_file:
42
43         insert_regex = re.compile('INSERT INTO `pagelinks` VALUES (.*)\n')
44         row_regex = re.compile('\"\"\"(.*),(.*),\"(.*)\",(.*)\"\"\"')
45         avro_content = []
46         MAX_AVROCONTENT_LENGTH = 1000000 # Constante   mettre   jour selon la capacit  de la machine
47         numero_output = 1
48
49         for line_bytes in sql_file:
50             #line = str(line_bytes, 'utf-8')
51             line = str(line_bytes, encoding="ISO-8859-1")
52             match = insert_regex.match(line.strip())
53             if match is not None:
54
55                 data = match.groups(0)[0]
56                 rows = data[1:-1].split("),(")
57                 for row in rows:
58                     row_match = row_regex.match(row)
59                     if row_match is not None:
60                         pl_from = row_match.groups()[0]
61                         pl_namespace = row_match.groups()[1]
62                         pl_title = row_match.groups()[2]
63                         pl_from_namespace = row_match.groups()[3]
64
65                         avro_content.append({"pl_from":int(pl_from), "pl_namespace":int(pl_namespace),
66
67                         if len(avro_content) > MAX_AVROCONTENT_LENGTH :
68                             avro_file = avro_path + '-' + str(numero_output).zfill(3) + '.avro'
69                             sauvegarder_liste(hdfs_client, avro_file, schema, avro_content)
70                             del avro_content # Lib rer la m moire
71                             avro_content = []
72                             numero_output += 1
73
74
75         print('Ecriture du dernier liste dans HDFS')
76         avro_file = avro_path + '-' + str(numero_output).zfill(3) + '.avro'
77         sauvegarder_liste(hdfs_client, avro_file, schema, avro_content)
78         print('Fin de la s rialisation, au revoir !!!')
```


3.b) Sérialisation avec avro: pagelinks (output séparé python) ~37h

```
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-pagelinks-324.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-pagelinks-325.avro
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-pagelinks-326.avro
Ecriture du dernier liste dans HDFS
Ecriture dans HDFS de : /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-pagelinks-327.avro
Fin de la sérialisation, au revoir !!!
```

```
real    2227m18.391s
user    2213m40.047s
sys      7m41.099s
```

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/data/frwiki/frwiki-20200201/master/full

Go!

Show 25 entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	22.26 MB	Jul 16 15:26	3	128 MB	frwiki-20200201-pagelinks-001.avro	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	20.94 MB	Jul 16 15:32	3	128 MB	frwiki-20200201-pagelinks-002.avro	
<input type="checkbox"/>	-rwxr-xr-x	maiky	supergroup	20.36 MB	Jul 16 15:37	3	128 MB	frwiki-20200201-pagelinks-003.avro	

4. Requêtes Spark SQL



5. Résultats:

Environ 8h pour exécuter les requêtes Spark SQL.

Avec ma machine:

- Lenovo ThinkPad-Edge-E330
- SE: Ubuntu 16.04 LTS
- Mémoire: 7,4 Gio
- Procésseur: Intel® Core™ i5-3210M CPU @ 2.50GHz x 4

Il a fallut environ :

- 8 heures pour la sérialisation de l'history pages (75,9 Go)
- 37 heures pour la sérialisation de la pagelinks (11,9 Go)
- 8 heures pour interroger les meilleurs contributeurs avec Spark SQL

Voici les meilleurs contributeurs :

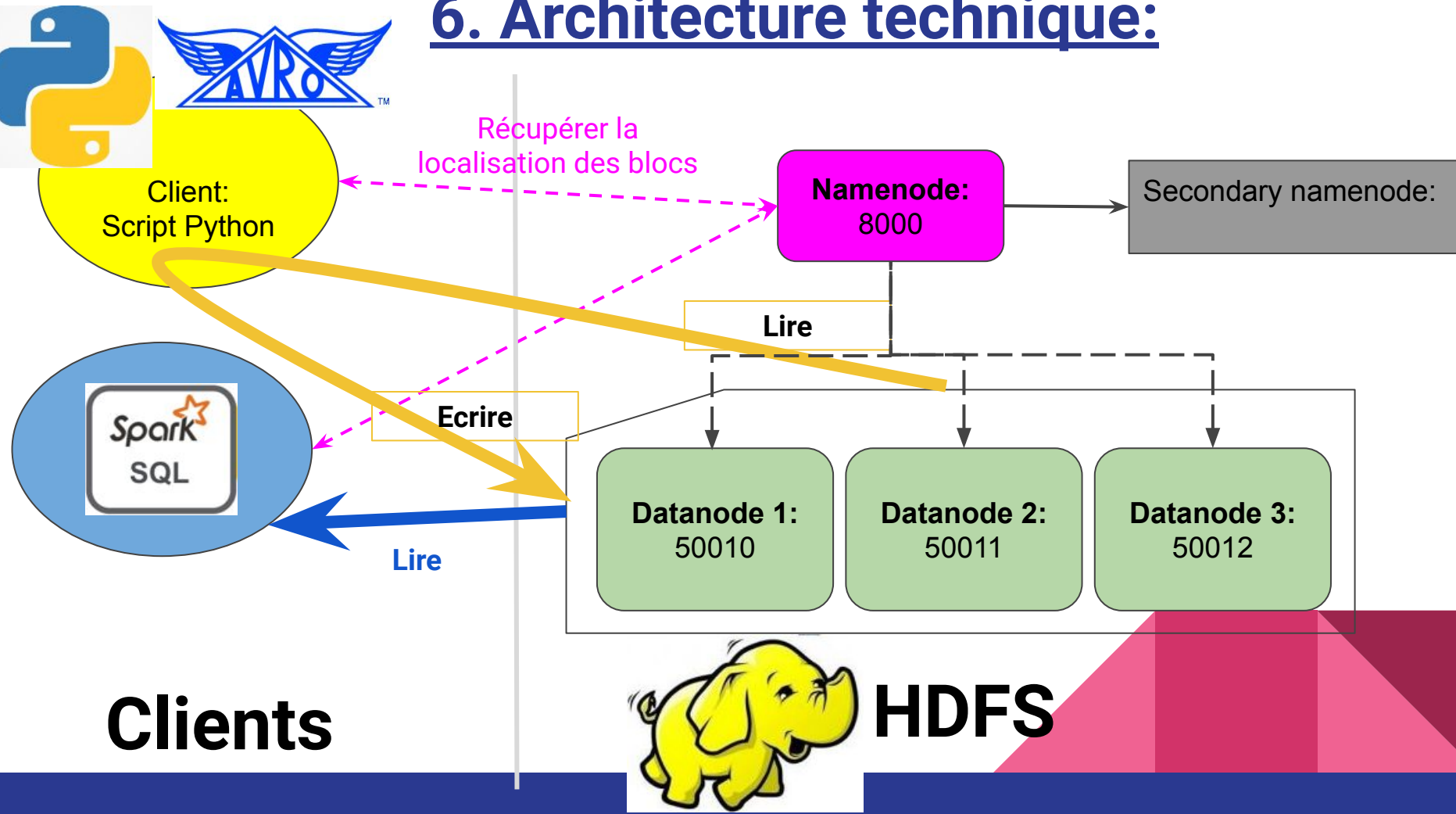
name	id	ip	count
Léon66	100556	null	234
Jolek	862133	null	205
Pierrette13	1220016	null	133
Documentation-F.R...	240530	null	92
G de gonjasufi	964044	null	80
null	null	145.232.230.254	74
Cerhab	431962	null	72
Lylvic	210569	null	64
null	null	89.224.58.251	60
Bruinek	322798	null	50
Yugiz	57049	null	44
Buddho	29236	null	42
Cantons-de-l'Est	502332	null	41
null	null	81.62.239.52	37
Perky	71065	null	34
Foudebassans	1591710	null	31
Francois Torrelli	3082177	null	29
Puckstar	825581	null	27
Jean-Louis Swiners	207174	null	27
HerculeBot	346772	null	24

only showing top 20 rows

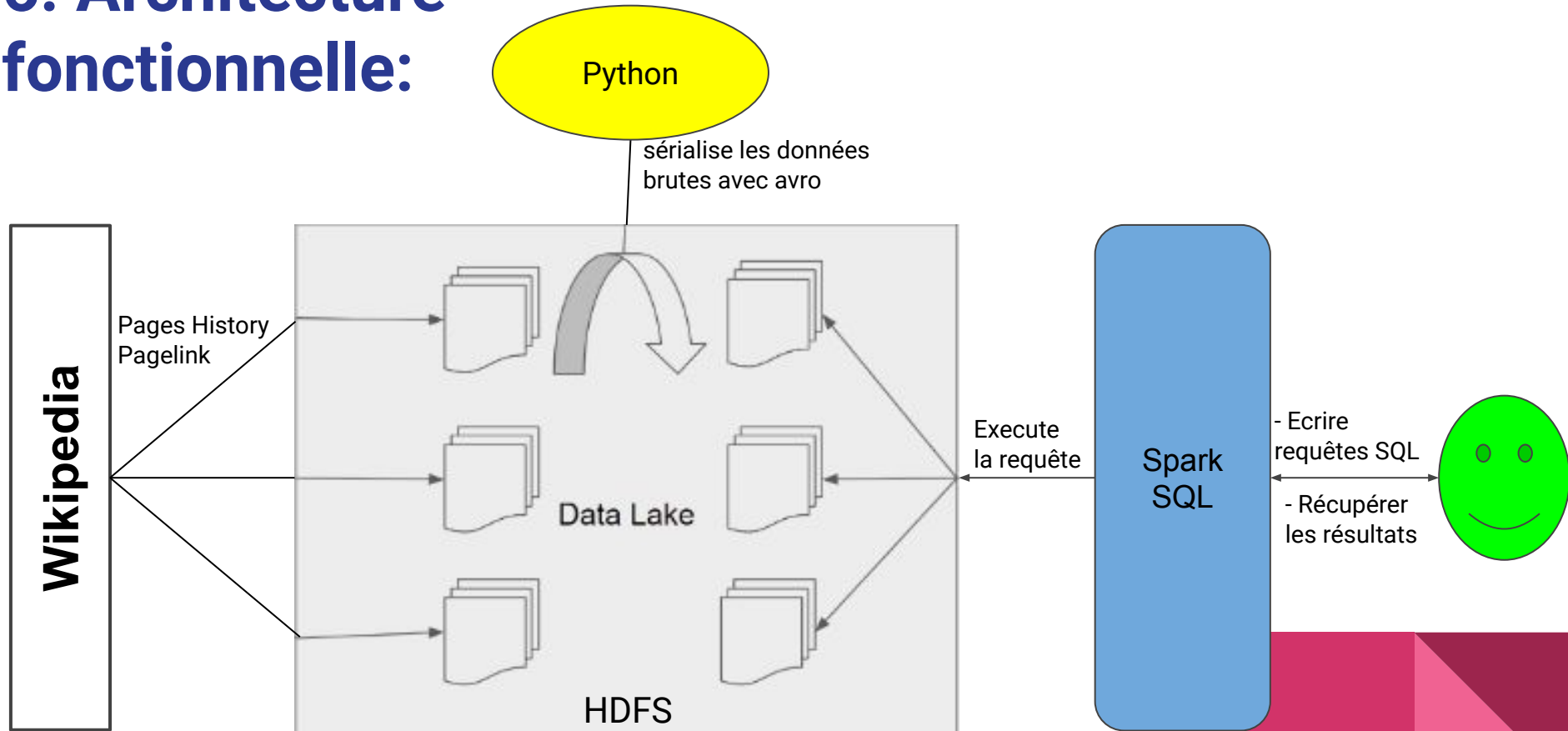
***** FIN *****

real 483m45.890s
user 91m53.195s
sys 6m55.138s

6. Architecture technique:



6. Architecture fonctionnelle:



7. Conclusion:

1. On a vu que pendant la mise en place d'un data lake, il faut prévoir tout:
 - a. scénarios catastrophes
 - b. pannes de machine
 - c. utilisation et évolution des données dans le temps

2. Optimisation du script de sérialisation est crucial!!
 - Éviter d'encombrer mémoire pour le traitement sinon pas de données sérialisé
 - Traiter par lots
 - code optimisé peut faire de gagner beaucoup de sérialisation

3. Requêtes sur de gros volumes de données très coûteuse en temps:
 - a. Intéressant d'optimiser le nombre de machines noeuds
 - b. Utiliser *[persist\(\)](#)* avec stockage level si nécessaire pour éviter de répéter les requêtes dans HDFS à chaque fois

Annexe



5. Résultats:

4.5. Tri et présentation du résultat:

```
contributor_count_df = contributor_df.groupby(["name", "id", "ip"]).count().orderBy("count", ascending=False)
contributor_count_df.show()
```

	name	id	ip	count
	Léon66	100556	null	234
	Jolek	862133	null	205
	Pierrette13	1220016	null	133
	Documentation-F.R...	240530	null	92
	G de gonjasufi	964044	null	80
	null	null	145.232.230.254	74
	Cerhab	431962	null	72
	Lylvic	210569	null	64
	null	null	89.224.58.251	60
	Bruinek	322798	null	50
	Yugiz	57049	null	44
	Buddho	29236	null	42
	Cantons-de-l'Est	502332	null	41
	null	null	81.62.239.52	37
	Perky	71065	null	34
	Foudebassans	1591710	null	31
	Francois Torrelli	3082177	null	29
	Puckstar	825581	null	27
	Jean-Louis Swiners	207174	null	27
	HerculeBot	346772	null	24

only showing top 20 rows

2.a) Description de stockage: Structure de notre data lake

- /data/ : Contient tous nos données
- /data/.snapshot/s20200720-163848.488/* : contient le snapshot de tous nos données
- /data/frwiki/ : contient tout ce qui provient de Wikipedia
- /data/frwiki/raw/ : contient tous nos données brutes
- /data/frwiki/raw/frwiki-20200201-pagelinks.sql : fichier brute contenant les liens entre les pages (11.04 GB)
- /data/frwiki/raw/frwiki-20200201-stub-meta-history.xml : fichier brute contenant les informations et historique des pages (70.72 GB)
- /data/frwiki/frwiki-20200201/ : contient tout ce qui provient des données brutes du 1er février 2020
- /data/frwiki/frwiki-20200201/master/ : contient tous les jeux de données provenant des données brutes wikipédia du 1er février 2020
- /data/frwiki/frwiki-20200201/master/pagelink.avsc : schéma avro utilisé pour sérialiser la pages link
- /data/frwiki/frwiki-20200201/master/pageshistory.avsc : schéma avro utilisé pour sérialiser l'historique et les informations concernant les pages
- /data/frwiki/frwiki-20200201/master/full/ : contient tous les données sérialisé (complet)
- /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-pagelinks-*.avro: pages links sérialisés
- /data/frwiki/frwiki-20200201/master/full/frwiki-20200201-stub-meta-history-00520.avro : history sérialisé
- /data/frwiki/frwiki-20200201/master/test : contient des échantillons de jeux de données sérialisés ==> pour faire des tests de traitement ou d'observer des de nos données