Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

TIC – Filière Télécommunications

Orientation Réseaux et Sécurité

Projet de Bachelor

10 juillet 2015

Utilisation du NoSQL pour l'analyse des données du NetObservatory

Rapport final

Version 1.0

Étudiant :	David Rossier
Mandaté par :	NetObservatory
Supervisé par :	François Buntschu Houda Chabbi Drissi
Experts :	Pierre-Alain Mettraux Robert Van Kommer

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Historique du document

Version	Auteur	Description des modifications	Date
0.1	David Rossier	Création du document	27.05.2015
0.2	David Rossier	Ajout des chapitres et mise en forme document	02.07.2015
0.3	David Rossier	Logo et numéros des chapitres Chapitres : Conception et Implémentation	06.07.2015
0.4	David Rossier	Résumé / Abstract	06.07.2015
0.5	David Rossier	Complément Analyse : Scala Conception : Revue Implémentation : Chapitres importation et organisation des fichiers Glossaire Tests fonctionnels	07.07.2015
0.6	David Rossier	Corrections d'orthographe Ajout des sections 4.5 à 4.7 de l'implémentation Conclusion	08.07.2015
0.7	David Rossier	Abstract Contenu CD	09.07.2015
0.8	David Rossier	Conclusion, compléments Spark & Scala Bonnes pratiques Tests de performance Perspectives, Conclusion, problèmes rencontrés	09.07.2015
0.9	David Rossier	Corrections	10.07.2015
1.0	David Rossier	Finalisation	10.07.2015

Résumé

Dans un monde qui entre dans l'ère des données, le BigData devient de plus en plus présent dans tous les domaines. Les données récoltées au fil du temps deviennent une source d'analyses profondes, par exemple pour mieux comprendre les habitudes humaines à travers l'apprentissage par la machine. De nouvelles technologies et de nouvelles infrastructures deviennent nécessaires pour traiter ces grandes quantités de données. Ces nouvelles technologies utilisent des architectures distribuées, tant pour stocker que pour traiter les données.

Créé par la fondation Apache sur la base de documents publiés par Google sur leur implémentation BigTable, Map/Reduce et Google FS, l'écosystème Hadoop offre un système de fichiers distribués avec HDFS et plusieurs applications basées sur les Map/Reduce permettant de simplifier l'implémentation de scripts parcourant ces données dans le but de les analyser.

Le projet de recherche NetObservatory, développé depuis 2009 à la Haute école d'ingénierie et d'architecture de Fribourg regroupe plusieurs travaux de Bachelor et de Master. Ce projet a généré plusieurs outils qui collectent régulièrement des données sur la sécurité des infrastructures Internet suisses, notamment sur les services Web et DNS. Ces données sont stockées dans des bases de données MySQL et représentent chaque année plus d'une centaine de Giga-octets de données. Elles ont pour but d'être analysées sous forme de plusieurs graphiques dans le NetObservatory Report & Analysis (NORA), mais peuvent également être utilisées pour effectuer des recherches sur de nouvelles failles qui pourraient apparaître. La génération de ces graphiques peut prendre beaucoup de temps, et les bases de données MySQL actuelles semblent atteindre leurs limites lors de requêtes sur de grosses quantités de données.

Le projet NODaBOp (Utilisation du NoSQL pour l'analyse des données du NetObservatory) étudie différentes solutions proposées par l'écosystème Hadoop et propose la modélisation d'une nouvelle architecture pour traiter les données du NetObservatory à l'aide de cet écosystème. La migration des données et la création de plusieurs scripts de génération de graphique ont pu être intégrés à un système Hadoop, validant ainsi la possibilité de réaliser une telle migration. Les tests de performance sur les deux architectures ont permis de démontrer que les requêtes travaillant sur de grandes quantités de données obtiennent de meilleurs résultats dans l'écosystème Hadoop, bien que les requêtes travaillant sur de petites quantités de données soient plus rapides sur le système MySQL actuel.

Abstract

In a world that enters the data era, the area of Big Data becomes more and more present in every single domain. The data gathered over time become a source of deep analysis, for example to better understand people, through machine learning. New technologies and new infrastructure become necessary to handle these large amounts of data. These new technologies use distributed architectures, both for storage and for data processing.

Created by the Apache Foundation on the basis of documents published by Google on their BigTable, Map / Reduce and Google FS implementations, and used by Facebook and Yahoo for datamining, the Hadoop ecosystem provides a distributed file system with HDFS and several applications based on Map / Reduce to simplify script implementation to access data in order to analyze them.

The NetObservatory research project, developed since 2009 at the University of Applied Sciences and Arts of Fribourg, includes several Bachelor and Master works. This project has generated several tools over time that regularly collect data on the security level of the Swiss Internet infrastructure, including Web services and DNS. This data is stored in MySQL databases and represent annually more than a hundred gigabytes of data. These data are then analyzed and produce some statistics represented as multiple graphics in the NetObservatory Report & Analysis (NORA), but can also be used to conduct research on new vulnerabilities that might appear. The generation of these statistics can be time consuming, and current MySQL databases seem to reach their limits when the queries are on large amounts of data.

The project “NODaBOp” (Utilisation of NoSQL for NetObservatory’s data analysis) studies different solutions proposed by the Hadoop ecosystem and proposes the modeling of a new architecture to handle NetObservatory’s data. The data migration and the development of some statistics scripts were integrated to an Hadoop system, proving that it is possible to realize such a migration. Performance tests on both architectures show that the applications working on large amounts of data achieve better results in the Hadoop ecosystem, while queries working on small amounts of data are faster on the current MySQL system.

Table des matières

1. Introduction

La sécurité des infrastructures informatiques dépend de plusieurs facteurs, soit de la configuration du réseau et des éléments de protection tels que les firewalls, mais surtout de la fiabilité des applications utilisées par les serveurs et les clients (ainsi que des systèmes d'exploitation). Tous les logiciels du marché présentent des vulnérabilités qui peuvent être exploitées de façon malveillante par des hackers, soit pour voler de l'information, soit pour déployer des réseaux de bots.

Ces infrastructures sont le plus souvent connectées à Internet et nécessitent un nom de domaine pour pouvoir fonctionner. Dans le cadre du projet de recherche NetObservatory mené à la HEIA-FR depuis 2009, différentes données sont collectées sur le réseau Internet suisse, telles que le type et les versions logicielles utilisées par les serveurs web. Diverses statistiques sont ensuite générées comme, par exemple, le top 10 des outils web (CMS) les plus utilisés ou encore le top 10 des domaines les plus vulnérables.

Le projet NetObservatory a permis de développer des outils Python de récolte de données sur le réseau Internet suisse dans le cadre de plusieurs projets de bachelor et de master. Comme on peut le voir dans la Figure 1, ces scripts enregistrent les données récoltées dans des tables MySQL, en indiquant la date de la récolte. Les données collectées représentent aujourd'hui plus de 900GB d'informations.

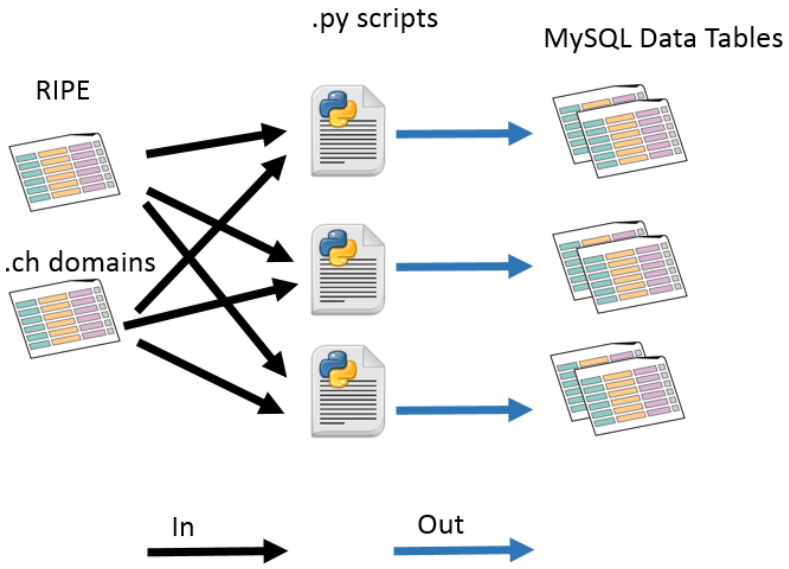


Figure 1 : Récolte de données

Ces données sont ensuite utilisées pour générer des graphiques de statistiques, qui sont utilisées principalement à l'heure actuelle pour la génération du rapport NORA (NetObservatory Report & Analysis). Le rapport NORA permet d'évaluer le niveau de sécurité offert par les infrastructures suisses et était publié plusieurs fois par année jusqu'en 2013. Ces statistiques sont générées à l'aide de scripts Python qui interrogent la base de données MySQL et génèrent des graphiques (Figure 2).

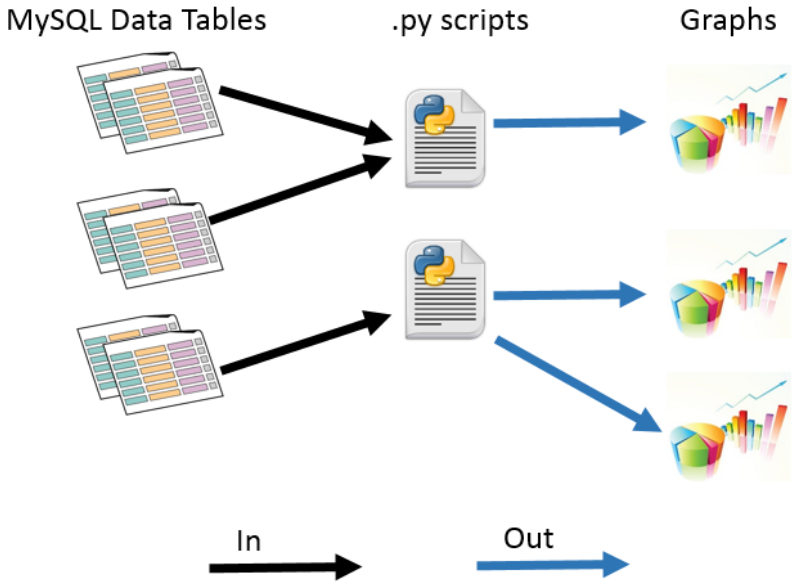


Figure 2 : Génération des graphiques

La génération des statistiques au travers des requêtes sur la base de données devient très coûteuse en temps et en ressources. De plus, les mandants du projet aimeraient passer à des générations de données plus fréquentes, ou à la demande. Cela représentera donc d'avantage de données et l'architecture MySQL actuelle semble atteindre ses limites.

Le but de ce projet est de proposer une migration de cette base de données MySQL vers une base de données NoSQL afin d'optimiser l'utilisation de ces données. Les objectifs principaux du projet sont l'apprentissage et la découverte de l'écosystème Hadoop, la proposition d'une intégration d'Hadoop dans le projet NetObservatory et une comparaison des performances entre le nouveau processus et l'ancien.

Le cahier des charges et la planification de ce projet sont disponibles dans l'annexe 1 de ce document.

Nous analyserons tout d'abord l'état de l'art, tant sur les différents éléments du projet NetObservatory existant que sur l'écosystème Hadoop, puis nous aborderons la conception, détaillant les différentes méthodes d'intégration possibles entre Hadoop et NetObservatory. Le chapitre d'implémentation détaillera les problèmes rencontrés lors du projet et les contraintes et conseils d'implémentation importants à connaître afin de reprendre la suite du projet. Finalement, des tests seront réalisés afin de valider le fonctionnement, et des tests de performance sur le cluster DAPLAB (Data Analysis and Processing Lab), cluster Hadoop installé à la HEIA-FR permettront de comparer les temps d'exécution sur les différentes architectures.

2. Analyse

2.1. NetObservatory

L'architecture actuelle du NetObservatory fonctionne à l'aide de deux types d'outils :

- Récoltes de données sur le réseau Internet suisse
- Statistiques générées à l'aide des données récoltées

Les données récoltées sont actuellement enregistrées dans des tables MySQL.

Dans le cadre du projet, il sera important de comprendre les relations entre les différents éléments.

2.1.1. Outils de récoltes

Les outils de récolte sont des scripts Python qui se contentent de récolter des données sur l'Internet suisse et de les enregistrer dans différentes tables MySQL.

L'annexe 2 de ce document recense les tables utilisées par chacun des scripts Python, en entrée comme en sortie.

2.1.2. Outils de statistiques

Les outils statistiques du NetObservatory sont des scripts Python qui questionnent la base de données du NetObservatory pour ensuite en tirer des graphiques.

L'annexe 3 de ce document recense les tables utilisées par chacun des scripts, ainsi que les projections (valeurs récupérées dans ces tables) ainsi que les détails importants de chacune des requêtes effectuées, tels que les jointures et les filtres.

De plus, l'annexe 4 de ce document indique si les scripts Python et les requêtes MySQL sont écrites de manière adaptée pour l'architecture actuelle, et propose quelques pistes d'améliorations.

2.1.3. Contenu de la base de données

La base de données `archive` du NetObservatory contient toutes les tables utilisées par les différents outils cités précédemment.

Il est important de déterminer la taille que prendra une itération des scripts de récolte. Le tableau ci-dessous recense la taille de la dernière version des tables du NetObservatory. Elle sera utilisée à titre indicatif, bien qu'il soit évident que la quantité de données évoluera au fil des années.

Table	Size [MB]	Index [MB]	Total [MB]
1ld_20140430	317,19	2,38	319,57
2ld_20140430	3031,86	42,32	3074,18
3ld_20140430	0	0	0
annonceV4_20150509	1,52	4,03	5,55
annonceV6_20150509	0,14	0,25	0,39
autoSystem_20150509	0,44	NA	0,44
bgpresults_20150509	0,02	NA	0,02
dnssec_validation_20140430	151,13	42,59	193,72
dns_20150525	244,91	421,6	666,51
dns_discover_20150406	913,72	117,81	1031,53
lookingGlass_20130208	0,02	NA	0,02
npa_20110224	0,16	0,12	0,28
nvd_cvss_20150406	4,12	1,18	5,3
nvd_vulns_20150406	88,6	20,26	108,86
reverse_dns_20150518	806,26	56,48	862,74
ripe_20150519	1,59	1,27	2,86
root_20120430	0,05	0,01	0,06
routeIPv4_20150509	1,52	NA	1,52
routeIPv6_20150509	0,13	NA	0,13
scan_os_20150501	12,32	0	12,32
scan_services_20150501	22,71	0	22,71
smtp_discover_20150413	13,05	0,92	13,97
ssl_discover_20150504	1560,94	130,11	1691,05
voisins_20150509	2,51	1,52	4,03
web_fingerprint_20150522	75	98,54	173,54
web_fingerprint_full_20150522	3000	NA	3000
whois_20110201	252,02	54,17	306,19
Total [MB]	**10501,93**	**995,56**	**11497,49**
Total [GB]	**10,25579102**	**0,972226563**	**11,22801758**

Comme on peut le constater, la quantité de données récoltées représente 11.2 GB par mois, soit environ 135 GB de données par année. En imaginant une extension des outils de récolte à d'autres régions, ou pays, comme par exemple l'Europe, cela représenterait des centaines de GB par mois. De plus, de nouveaux outils de récolte pourront être développés dans le cadre du projet NetObservatory, ce qui représenterait d'avantage de données. L'utilisation d'une nouvelle architecture, telle que celle offerte par l'écosystème Hadoop, est donc justifiée.

Les annexes 5 et 6 contiennent d'avantages de détails sur les champs de chacune des tables, ainsi que l'évolution de la taille des tables dans le temps.

2.2. Hadoop

L'écosystème Hadoop est un Framework Java libre créé par Doug Cutting. Il a été développé par la fondation Apache dans le but de traiter de grandes quantités de données de manière distribuée et échelonnable. Inspiré par les publications MapReduce, Google FS et BigTable de Google, il en reprend les concepts pour fournir un environnement Open Source pour les Big Data.

Hadoop est implémenté au-dessus du Framework Java, il est donc multiplateforme.

L'écosystème Hadoop est très récent et évolue très rapidement. Ainsi, la première version de l'architecture utilisait uniquement le principe des Map/Reduce pour exécuter les requêtes.

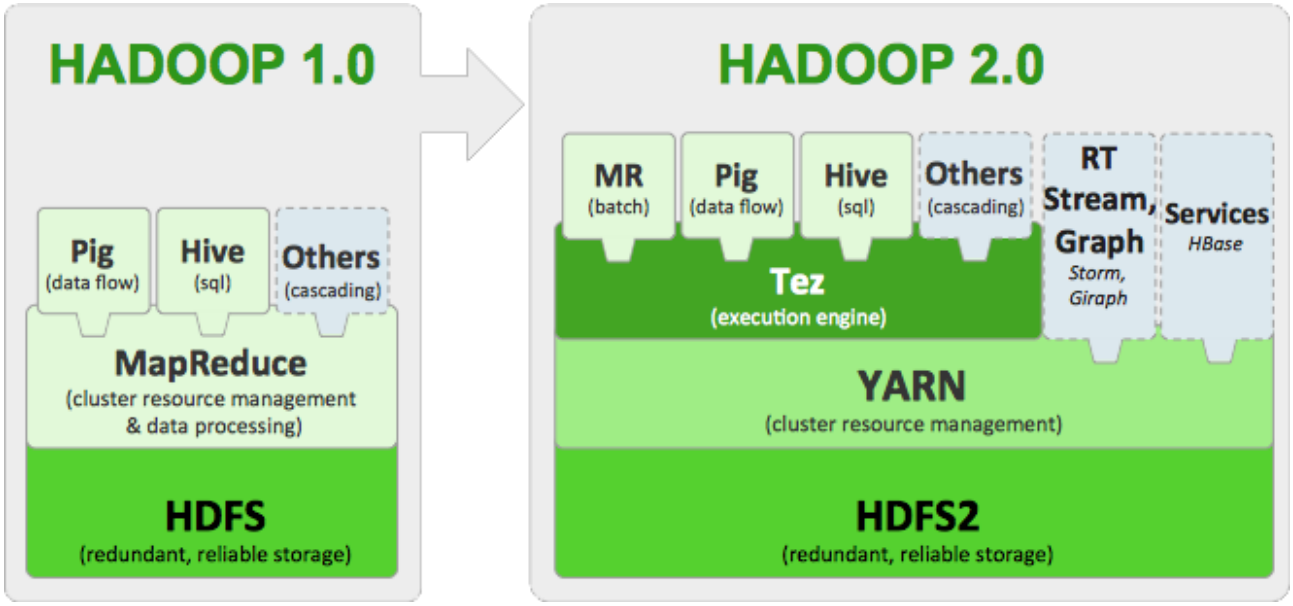


Figure 3 : Evolution Hadoop [1]

La seconde version d'Hadoop offre une nouvelle interface d'abstraction gérant les ressources de travail. Cette interface d'abstraction se nomme YARN. Ainsi, toutes les tâches utilisant les ressources d'un cluster Hadoop sont gérées par le gestionnaire de ressource YARN.

Hadoop est séparé en deux parties différentes et indépendantes : le stockage (HDFS) et le calcul (YARN).

Les prochains chapitres détaillent le fonctionnement et les tâches effectuées par chacune des parties du système, ainsi que les outils les plus connus de l'écosystème Hadoop.

2.2.1. HDFS

Le système de fichiers HDFS (Hadoop Distributed File System), écrit en Java et utilisé par Hadoop a été créé dans le but de transformer un cluster de serveurs habituel en un outil de stockage massif. HDFS a été particulièrement créé dans le but d'obtenir une architecture « Write-once, read-many », c'est-à-dire qu'une fois les données écrites, elles ne devraient pas être modifiées.

Créé spécialement pour une utilisation avec de grandes quantités de données, où l'extensibilité, la fiabilité et la vitesse de traitement sont critiques, HDFS accepte tous les formats de données.

La particularité de HDFS par rapports aux autres systèmes de fichiers distribués est qu'il a été créé de manière à fonctionner sur du matériel bon marché.

Une instance du système HDFS peut représenter un cluster de centaines ou de milliers de serveurs, chacun stockant une partie des données du système. Cela représente un gigantesque nombre de composants matériels, avec pour chacun un risque de problèmes. Cela signifie que certains composants de l'architecture HDFS peuvent être en permanence en panne. C'est pourquoi la détection des erreurs et une restauration rapide du système sont les buts premiers d'HDFS.

HDFS et Hadoop étant deux systèmes relativement récents, certaines des fonctionnalités souhaitées ne sont pas encore complètement implémentées. Les chapitres suivants nous permettront de découvrir les fonctionnalités les plus importantes du système HDFS ainsi que son fonctionnement.

Architecture HDFS

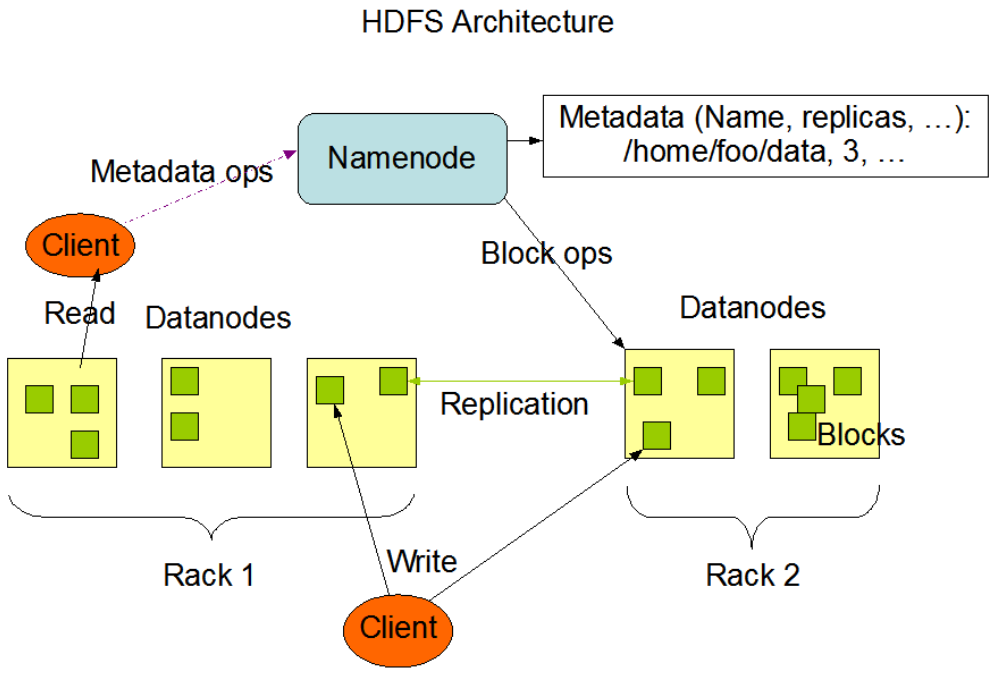


Figure 4 : HDFS Architecture NameNode et DataNode [2]

Comme on peut le voir dans la Figure 4, l'architecture HDFS est divisée en deux points principaux : Le **NameNode** et les **DataNodes**. Le NameNode est une sorte de contrôleur de l'architecture. Il contient l'espace de nom, l'arborescence du système de fichier et les métadonnées des fichiers et répertoires, qui permettent de savoir où se trouve chacune des données sur le cluster.

Un NameNode est unique dans l'architecture HDFS mais peut disposer d'une instance secondaire qui gère l'historique des modifications dans le système de fichier. Cette instance a un rôle de backup et permet donc une redondance du NameNode en cas de panne. Néanmoins, toutes les

informations du namespace étant stockées en RAM, il n'est pas exclu que certaines informations soient perdues en cas de problème du NameNode.

Les DataNodes sont les composants de l'architecture sur lesquels sont stockées les données réelles. Ils gardent le contact avec le NameNode en envoyant régulièrement des messages « **Heart beat** » (battement de cœur) qui confirment leur présence, ainsi que des messages « **Block Report** » indiquant les différents blocs stockés sur le DataNode.

Stockage et réplication

Lorsqu'un fichier doit être stocké dans HDFS, c'est le NameNode qui est contacté. Le fichier à écrire est alors divisé en plusieurs blocs de données – la taille des blocs peut-être configurée et est généralement de 128MB. Tous les blocs d'un même fichier sont de la même taille, sauf le dernier. Les blocs de données seront répartis sur les différents DataNodes du cluster. Le nombre de réplicas est appelé « **Facteur de réplication** ». Le facteur de réplication et la taille des blocs peuvent être configurés différemment pour chaque fichier. Un facteur de trois est recommandé pour une résilience aux pannes. La réplication des blocs permet une redondance des données et permet donc d'éviter des pertes de données en cas de problèmes matériels sur l'un des éléments du cluster. Le schéma de la Figure 5 résume la manière dont sont stockés les blocs sur le cluster.

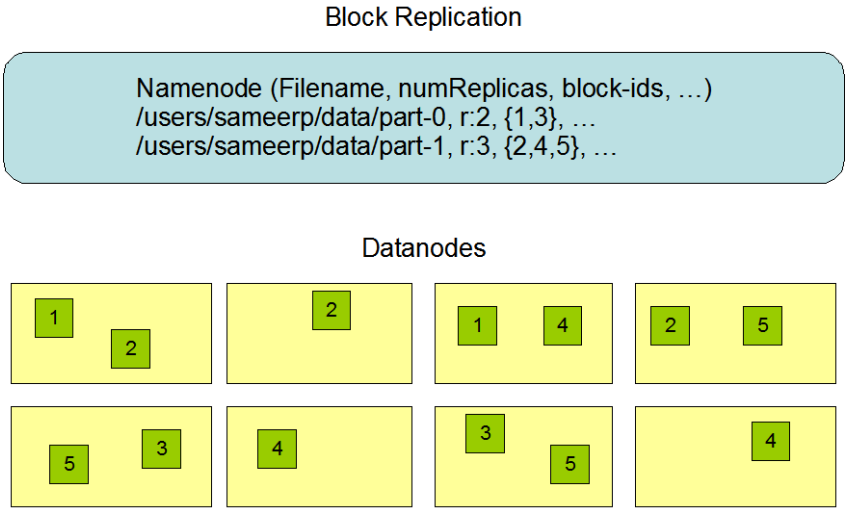


Figure 5 : Réplication des blocs de données [3]

Le NameNode s'occupe activement de vérifier si le nombre de réplicas pour chacun des blocs est suffisant. Ainsi, lors d'un problème sur un des DataNode découvert à l'aide des « HeartBeat », il se charge de créer d'autres réplicas des blocs qui y étaient stockés. Néanmoins, il ne s'occupe pas directement de la copie des blocs. Lorsqu'il se rend compte qu'un bloc doit être répliqué, il répond à un message « Heart beat » en envoyant une instruction au DataNodes indiquant qu'un bloc stocké sur son disque doit se répliquer. C'est alors le DataNode qui se charge de la réplication du bloc sur le ou les autre(s) DataNodes.

Il est important de noter qu'il est possible d'indiquer la topologie physique des différents nœuds du cluster. Ceci se fait à l'aide de l'élément de configuration « `topology.node.switch.mapping.impl` ». Cet outil permet de mapper des adresses IP à un emplacement de « rack » physique. La topologie est utilisée pour optimiser tant la réplication que les interactions entre les données lors d'un Map/Reduce.

La position des réplicas est critique dans HDFS pour la disponibilité et la performance. Optimiser le positionnement des réplicas est l'une des fonctionnalités clés de HDFS. Le but est d'améliorer au mieux l'accessibilité, la fiabilité et l'utilisation de la bande passante du réseau. L'optimisation de cette fonctionnalité n'en est encore qu'à ses débuts et nécessitera des améliorations dans le choix des nœuds pour chaque bloc.

Un autre point-clé d'HDFS est qu'il déplace le traitement de données sur HDFS, et non inversement. Les tâches de traitement peuvent donc s'exécuter sur le nœud physique où résident les données, ce qui réduit le trafic sur le réseau. Ainsi, le choix du réplica dans le cluster dépend

de la position du client par rapport au cluster : c'est le bloc (partie d'un fichier) le plus proche du client qui sera choisi comme source des données.

Démarrage et Safemode

Au démarrage, le NameNode entre dans un état spécial nommé « Safemode ». Aucune réplication n'est effectué lorsque le NameNode est dans cet état. Le NameNode écoute les messages HeartBeat et BlockReport des DataNodes. Un BlockReport contient la liste des blocs de données que le DataNode héberge. Chaque bloc a un nombre minimum de réplicas. Après avoir reçu un pourcentage de blocs correctement répliqués (le nombre minimum de réplicas est atteint), le NameNode quitte le Safemode et s'occupe de faire répliquer les blocs qui n'ont pas le nombre de réplicas spécifié sur d'autres DataNodes.

Persistance des métadonnées du système

L'espace de noms HDFS est stocké sur le NameNode. Chaque changement des métadonnées du système de fichiers est enregistré dans un log de transaction nommé « **EditLog** ».

Les changements enregistrés dans l'EditLog sont les suivants :

- Création, déplacement, suppression d'un fichier / dossier
- Modification du facteur de réplication

La totalité de l'espace de nom, incluant le mapping bloc-fichier et les propriétés des fichiers sont stockés dans un fichier nommé « **FsImage** ».

Les fichiers `EditLog` et `FsImage` sont enregistrés dans le système de fichier du système d'exploitation (non HDFS) du NameNode.

Le NameNode conserve une image de tout l'espace de nom du système de fichier, incluant le mapping bloc-fichier dans la mémoire RAM. Ces données-clés doivent être compactes de manière à ce qu'un NameNode ayant 4GB de RAM puisse supporter une énorme quantité de fichiers et de dossiers.

Lors du démarrage du NameNode, il lit l'image `FsImage` et le fichier `EditLog` depuis le disque. Il applique ensuite les transactions de l'EditLog dans la représentation en mémoire (RAM) de la FsImage et sauvegarde ensuite cette nouvelle image sur le disque, créant ainsi un « checkpoint ».

Dans l'implémentation actuelle, le checkpoint est créé uniquement au démarrage du NameNode. Dans le futur, cette sauvegarde devrait pouvoir être périodique.

Néanmoins, s'il existe un NameNode *secondaire*, celui-ci reçoit régulièrement (P.ex. toutes les heures) les modifications effectuées sur l'`EditLog`. Contrairement au NameNode principal, il ne reçoit pas de requêtes et peut donc effectuer des checkpoints plus réguliers de l'image disque. Cela peut offrir une récupération de l'état de l'image FsImage plus rapide.

Un DataNode enregistre les blocs de données HDFS dans son système de fichier local. Le DataNode n'a aucune connaissance sur les fichiers HDFS. Chaque bloc de données est enregistré dans un fichier séparé de son système de fichier.

Le DataNode ne crée pas tous les fichiers « blocs » dans un même répertoire. À la place, il utilise une méthode heuristique pour déterminer un nombre optimal de fichiers par dossiers et s'occupe de créer les dossiers de manière appropriée. En effet, il n'est pas optimal de créer tous les fichiers locaux dans un seul dossier car le système de fichiers ne sera peut-être pas capable de supporter un très grand nombre de fichiers dans un seul dossier.

Lors du démarrage d'un DataNode, celui-ci scanne son système de fichier, puis génère une liste de tous les blocs de données HDFS correspondants. Une fois terminé, le DataNode envoie un BlockReport au NameNode pour lui indiquer quelles données sont stockées sur son système de fichiers local.

Communication

Tous les protocoles de communication HDFS sont implémentés sur la couche TCP/IP. Un client établit une connexion TCP/IP sur un port configurable du NameNode. Il utilise le ClientProtocol

pour la communication entre le client et le NameNode. Les DataNodes utilisent le DataNode Protocol pour communiquer avec le NameNode.

Le DataNode Protocol et le ClientProtocol sont tous les deux encapsulés dans une Remote Procedure Call (RPC). Le NameNode n'initie jamais de RPC, mais peut répondre aux RPC ouvertes par les clients ou les DataNodes.

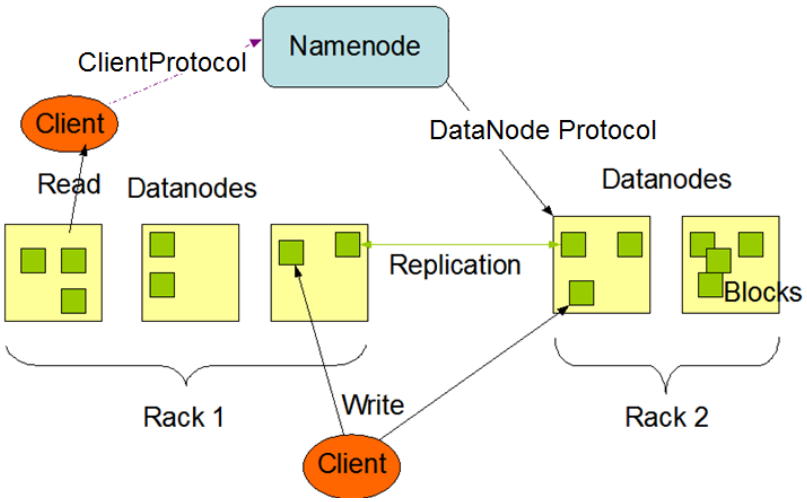


Figure 6 : Communication HDFS [2]

Robustesse

L'un des objectifs principaux de l'architecture HDFS est de stocker les données de manière fiable même en présence d'erreurs ou de problèmes de matériels ou de réseau.

HDFS offre plusieurs fonctions qui permettent de conserver en permanence la fiabilité et l'intégrité des données.

Réplication

En cas de problèmes causés par un problème de disque dur, une panne d'une partie du réseau, ou même une corruption des données, le NameNode se chargera de répliquer les blocs perdus pour que le facteur de réplication reste au moins à la valeur spécifiée. Les DataNodes n'ayant pas envoyé leurs messages Heartbeat sont considérés comme morts, et le NameNode n'effectuera plus aucune requête vers eux. Toutes les données qui y étaient stockées sont considérées comme non disponibles.

Equilibrage du cluster

L'architecture HDFS est compatible avec des schémas d''équilibrage automatique des données sur les nœuds du cluster. Ainsi, si l'espace libre minimal sur l'un des DataNodes atteint un certain seuil, les données pourraient être automatiquement déplacées sur un autre nœud.

De même, lorsqu'un fichier particulier subit une soudaine demande par les clients, un schéma pourrait automatiquement créer d'avantages de réplicas pour répartir la charge sur plusieurs DataNodes.

Malheureusement, ces schémas d'équilibrage ne sont **pas encore implémentés**.

Intégrité des données

Il est possible qu'un bloc de données récupéré d'un DataNode soit corrompu. Cela peut-être causé par des problèmes venant du matériel stockant les données, des problèmes de réseau ou encore des erreurs logicielles.

Le client HDFS implémente une vérification du checksum sur le contenu des fichiers HDFS. Ainsi, lorsqu'un client créé un fichier HDFS, il génère un checksum de chaque bloc du fichier et stocke ces checksums dans un fichier caché séparé dans le même espace de nom HDFS.

Lorsqu'un client récupère le contenu du fichier, il vérifie que les données reçues par chaque DataNode aient un checksum correct. Si ce n'est pas le cas, le client pourra récupérer le bloc de données depuis un autre DataNode qui a un réplica de ce bloc.

Problème de métadonnées

Les fichiers **`FsImage`** et **`EditLog`** sont les structures de données centrales de HDFS. Une corruption de ces fichiers peut rendre toute une instance HDFS non fonctionnelle. Pour cette raison, le NameNode peut être configuré pour maintenir plusieurs copies du FsImage et de l'EditLog. Chaque mise à jour de l'un ou l'autre des fichiers a pour conséquence une mise à jour des deux fichiers de manière synchrone. Cela réduit donc le nombre de transactions traitées chaque seconde. Néanmoins, cette dégradation est acceptable car les applications HDFS ne devraient pas avoir beaucoup d'accès aux métadonnées, mais plus aux données elles-mêmes.

Lorsqu'un NameNode redémarre, il choisit la dernière version des fichiers `FsImage` et `EditLog` à utiliser.

Utilisation HDFS sur Hadoop

Pour pouvoir déplacer des fichiers entre le système de fichier traditionnel et le système HDFS, il faut utiliser les commandes offertes par `hadoop fs` en ligne de commande :

```
hadoop fs -copyFromLocal <local-dir> <hdfs-dir>
hadoop fs -copyToLocal <hdfs-dir> <local-dir>
```

D'autres commandes existent également et sont les mêmes que pour le système Linux. Le format pour exécuter une commande est `hadoop fs command`.

`appendToFile, cat, checksum, chgrp, chmod, chown, count, cp, createSnapshot, deleteSnapshot, df, du, dus, expunge, find, get, getfacl, getfattr, getmerge, help, ls, lsr, mkdir, moveFromLocal, moveToLocal, mv, put, renameSnapshot, rm, rmdir, rmr, setfacl, setfattr, setrep, stat, tail, test, text, touchz, truncate, usage`

2.2.2. Map/Reduce

Map/Reduce est une méthode qui permet de distribuer une tâche sur plusieurs nœuds différents. Chaque nœud effectue les opérations de manière à ce que les données soient les plus proches possible des processeurs.

Une exécution d'un Map/Reduce consiste en différentes phases :

- Map
- Sort
- Shuffle
- Reduce

Les principaux avantages du niveau d'abstraction d'une tâche dans un Map/Reduce sont les suivants :

- Parallélisation automatique et répartition des données en blocs sur une infrastructure distribuée
- Tolérance aux pannes contre les problèmes de stockage, d'exécution ou d'infrastructure réseau
- Déploiement, monitoring et sécurité
- Abstraction propre pour les programmeurs

La plupart des programmes Map/Reduce sont écrits en Java, même s'ils peuvent également être écrits dans n'importe quel langage de Scripting en utilisant l'API Streaming de Hadoop.

Map/Reduce donne un niveau d'abstraction qui permet au développeur de se concentrer sur l'écriture des fonctions Map et Reduce.

Le schéma de la Figure 7 nous permet de mieux nous représenter le fonctionnement d'un Map/Reduce traditionnel.

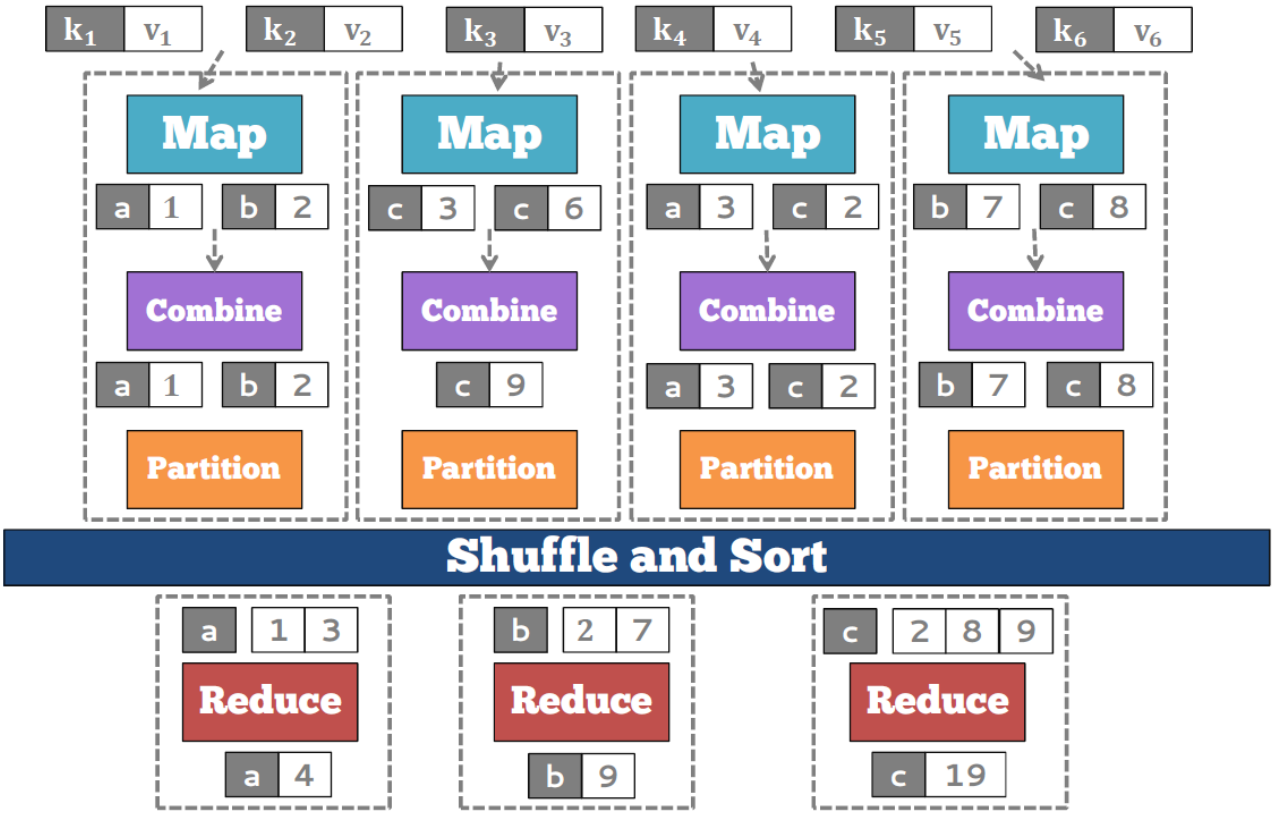


Figure 7 : Map / Reduce algorithme [5]

Le Mapper

Hadoop essaie de s'assurer que les tâches Mapper s'exécutent sur les nœuds qui contiennent localement la portion des données à traiter pour minimiser le trafic sur le réseau. Plusieurs Mappers s'exécutent en parallèle, chacun traitant une partie des données entrées.

Le Mapper lit les données sous forme de paires « clé/valeur ». Il retourne zéro ou plus de nouvelles paires « clé/valeur », qui peuvent avoir un format différent.

```
map(in_key, in_value) -> (inter_key, inter_value) list
```

Le Mapper peut utiliser ou complètement ignorer les clés d'entrées. En effet, dans certains cas, l'information contenue dans la clé n'est pas importante. Si le Mapper retourne quelque chose, la sortie doit être sous la forme de paires clé/valeur.

Le Reducer

Lorsque la phase « Map » est terminée, toutes les valeurs intermédiaires pour une clé intermédiaire donnée sont combinées dans une liste. Cette liste est envoyée à un Reducer. Un Job peut avoir un seul ou plusieurs Reducers, selon la configuration. Toutes les valeurs associées à une clé sont assurées d'arriver au même Reducer.

Les clés intermédiaires et leurs listes de valeurs sont passées au Reducer, triées selon l'ordre des clés. Cette étape est connue comme « Shuffle and sort ». Le Reducer retourne zéro ou plus paires de clé/valeurs finales, qui sont écrites dans HDFS. En pratique, le Reducer émet généralement une seule paire de clé/valeur pour chaque clé d'entrée.

Il est possible que certaines tâches Map prennent plus de temps pour se terminer que d'autres. Cela peut créer un goulet d'étranglement, car toutes les tâches Mappers doivent être terminées avant qu'un Reducer ne commence. Cela peut causer des ralentissements. Pour pallier à ce problème, si un Mapper apparaît comme s'exécutant beaucoup plus lentement que les autres, une nouvelle instance du Mapper sera lancée sur une autre machine, traitant le même set de données. Le résultat du premier Mapper terminant la tâche sera utilisé et les autres Mapper toujours en cours seront supprimés.

Streaming API

L'API Streaming permet aux développeurs d'utiliser n'importe quel langage pour écrire des Mappers et Reducers, tant que le langage est capable de lire le `stdin` et écrire dans le `stdout`.

L'avantage principal de cette API est qu'elle permet aux développeurs ne connaissant pas Java de pouvoir développer dans d'autres langages, et d'utiliser des librairies existantes.

Comment ça marche ?

Pour implémenter le streaming, il faut écrire séparément les programmes Mapper et Reducer dans le langage de son choix. Les scripts recevront l'entrée par `stdin` et écriront dans `stdout`.

Si le format `TextInputFormat` est utilisé, le Mapper streaming recevra simplement chaque ligne du fichier sur l'entrée `stdin` où aucune clé ne sera passée. Les sorties du Mapper et Reducer Streaming doivent envoyer leurs informations sous le format

```
key (tab) value (nouvelle ligne\n)
```

Un séparateur différent de la tabulation peut être spécifié.

En Java, toutes les valeurs associées à une clé sont passées au Reducer comme un itérateur. En utilisant Hadoop Streaming, le Reducer reçoit son entrée sous forme de paires (clé, valeur), une paire par ligne.

Le code devra garder trace de la clé pour qu'il puisse détecter lorsqu'une nouvelle clé démarre un Streaming Job.

Pour lancer un Streaming Job, il faut utiliser la commande suivante

```
hadoop jar $HADOOP_HOME/contrib/streaming/hadoop-streaming*.jar \
```

```
-input myInputDirs \ -output myOutputDir \
-mapper myMap.py \
-reducer myReduce.py \ -file myMap.py \ -file myReduce.py
```

L'écriture de bons Map/Reduce étant complexe, et les données stockées dans Hadoop devant souvent être traitées par des analystes qui ne connaissent pas forcément de bonnes méthodes pour les utiliser, des outils comme Hive et Pig ont été développés afin de fournir une certaine couche d'abstraction entre les données et les requêtes. Ces deux outils seront détaillés plus précisément dans les chapitres suivants.

Exemple de Map/Reduce

Un exemple de Map/Reduce est décrit dans l'annexe 7 de cette analyse.

2.2.3. Architecture YARN

La version 2 d'Hadoop apporte une nouvelle architecture.

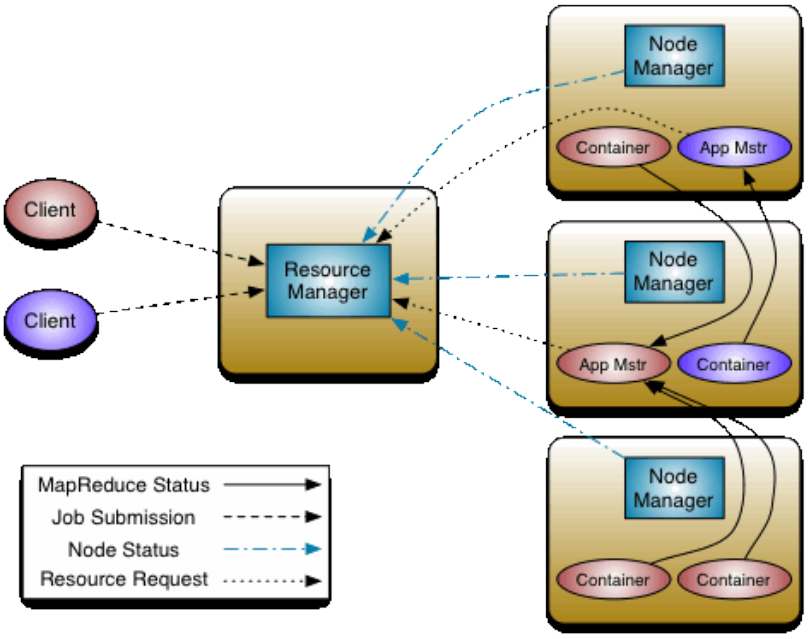


Figure 8 : YARN [4]

L'idée de YARN est d'avoir désormais un **ResourceManager** - l'autorité ultime pour arbitrer les ressources sur toutes les applications du système - et un **NodeManager** par nœud du cluster. Les NodeManager exécutent les tâches attribuées. Ils sont responsables des **Container** et se chargent d'informer le ResourceManager de l'utilisation des ressources.

Le ResourceManager se situe sur un nœud dit « Master ». Les clients envoient des « **Application** » au Resource Manager, qui assigne les ressources et transmet les tâches aux NodeManager.

Un **Job** est un programme capable d'effectuer l'exécution complète des Mappers et des Reducers sur un set de données. Une tâche est l'exécution d'un seul Mapper ou Reducer sur une seule part des données.

Le ResourceManager a deux tâches principales : **Scheduler** et **ApplicationsManager**.

Le **Scheduler** ne s'occupe que de la planification des tâches, et non de la surveillance de la réussite des tâches. Ainsi, il se base uniquement sur les besoins en ressources des applications, tant au niveau réseau, que performances cpu, mémoire ou disque. Pour ce faire, il utilise la notion abstraite de « **Container** ».

L'**ApplicationsManager** est responsable de la soumission des « Job ». C'est lui qui transforme le premier « Container » en **ApplicationMaster** et se charge de redémarrer l'ApplicationMaster en cas de panne.

Comme nous l'avons vu dans la Figure 3, toutes sortes d'applications peuvent fonctionner au-dessus de l'architecture YARN. Les prochains chapitres détaillent certains d'entre eux qui pourront être utilisés dans la partie d'implémentation de ce projet.

L'avantage principal de la nouvelle architecture YARN par rapport à l'ancienne des Map/Reduce est le nombre d'appels disque. En effet, dans la version 1 de Hadoop, une requête était séparée en plusieurs Jobs Map/Reduce, et le résultat de chacun d'entre eux était enregistré sur le disque HDFS. YARN offre la possibilité de supprimer ces accès disque et, de ce fait, d'améliorer les performances. On peut voir une schématisation de l'exécution d'une requête sur YARN par rapport à Map/Reduce dans la Figure 9.

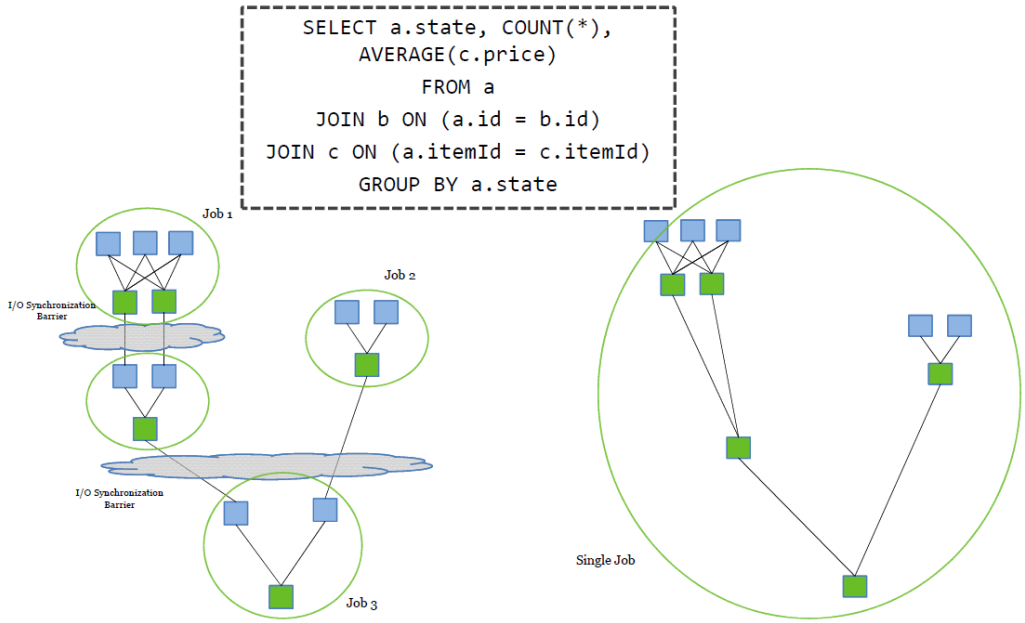


Figure 9 : Map/Reduce vs YARN [5]

2.2.4. Apache HCatalog

HCatalog est un outil de la fondation Apache qui fournit une couche d'abstraction entre les données et le stockage HDFS. Il permet de stocker les données principalement sous forme de tables. HCatalog supporte la lecture et l'écriture de fichiers de tous formats pour lesquels un SerDe (Serializer-Deserializer) peut être écrit. Il permet aux autres outils tels que Pig et Hive d'effectuer plus facilement des opérations sur les données.

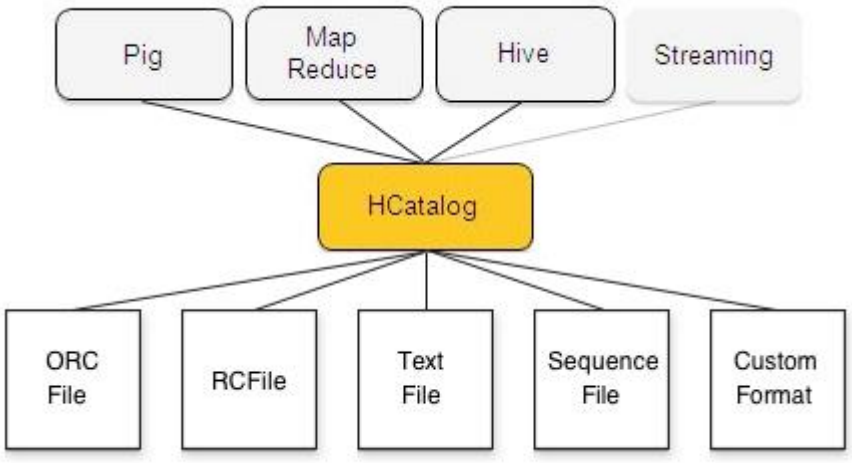


Figure 10 : HCatalog [6]

HCatalog offre une vue tabulaire des données. Néanmoins, il permet également d'offrir un partitionnement des données stockées. Ainsi, il est possible d'obtenir une partition par jour, par exemple.

2.2.5. Apache Hive

L'application Hive est une extension de Hadoop qui permet d'effectuer des requêtes avec un format de type SQL sur les données contenues sur HDFS. Elle génère des Map/Reduce sur la base des requêtes SQL effectuées et retourne les résultats demandés lors du Reduce final.

Actuellement Hive est capable de parser la plupart des commandes SQL, tant utilitaires (Show, describe, explain, set) que de création / suppression de tables (Create, partition, clustered, drop), ou de sélection de données, y compris les fonctions JOIN, GROUP BY et ORDER BY, mais également d'insertion et de suppression.

Dans le cadre du projet, c'est principalement la fonction de SELECT qui sera utilisée.

Note : Le format BLOB des colonnes MySQL n'est pas actuellement supporté par Hive. Néanmoins, il peut être utilisé en format String sous forme hexadécimale.

Stockage Hive

Les fichiers de Hive sont stockés dans la warehouse Hive. Il est possible de modifier les droits d'accès à une base de données Hive directement dans le système de fichier Hadoop.

Pour trouver l'endroit où est stockée la warehouse Hive, on peut utiliser la commande suivante :

```
hive> set hive.metastore.warehouse.dir;
hive.metastore.warehouse.dir=/apps/hive/warehouse
```

Cela permet de trouver l'endroit où sont stockées les tables, et, éventuellement, d'en modifier les droits d'accès pour empêcher d'autres utilisateurs d'y accéder.

Utilisation de Hive

Pour utiliser Hive sur un serveur Hadoop, il suffit de se connecter sur la Gateway principale et de lancer la commande `hive`.

Dès lors, il est possible d'exécuter une requête SQL dans la console hive.

Dans le cadre de ce projet, il sera néanmoins plus simple d'utiliser des requêtes pré-écrites et de les appeler à l'aide du paramètre –f.

Ainsi, pour lancer une commande SQL depuis un fichier, et sauvegarder le résultat de la commande dans un fichier, on peut utiliser la commande suivante :

```
hive –f path/to/sql/file > result.txt
```

2.2.6. Apache PIG

L'outil Apache Pig est une plateforme permettant d'analyser de grands sets de données, à l'instar de Hive. Si Hive est basé sur le format SQL, Pig est basé sur le langage Pig Latin, qui utilise une syntaxe proche des langages de Scripting.

La version actuelle de Pig fonctionne de manière similaire à Hive. Il s'agit d'un compilateur générant des programmes Map/Reduce de manière simplifiée en utilisant un langage de Scripting.

Par rapport à Hive, Pig permet d'effectuer les opérations les unes après les autres séquentiellement et d'avoir un contrôle complet sur chacune des étapes.

Les fonctions principales à noter et leurs syntaxes sont définies ci-dessous :

- REGISTER /path/to/jar/file/to/include
- LOAD 'path/to/file' USING PigStorage('SEP') AS (col1, col2, col3)
- FILTER dataset BY condition
- FOREACH dataset GENERATE …
- GROUP dataset BY col …
- ORDER dataset BY col1,col2
- STORE dataset INTO …
- JOIN dataset1 BY col, dataset2 BY col
- dataset1 UNION dataset2
- SPLIT dataset1 INTO dataset2 IF cond, dataset3 IF cond2
- DUMP(dataset)
- DISTINCT dataset
- LIMIT dataset nb

Il est également possible d'appeler des fonctions Java directement dans le script à condition qu'elles soient incluses au début à l'aide de la fonction Register, ce qui peut permettre d'effectuer des actions particulières sur les données.

2.2.7. Apache HBase

HBase est une base de données orientée colonnes. Cette terminologie étant proche de celles des SGBD (Systèmes de gestion de base de données) traditionnels, elle n'aide pas à mieux comprendre HBase. En réalité, une table HBase ressemble plus à une « map » multidimensionnelle.

Une *table* HBase est composée d'une multitude de *lignes*. Une *ligne*, dans le contexte de HBase, est une **clé** associée à une ou plusieurs *colonnes*. Les *lignes* sont automatiquement triées par ordre alphabétique. Pour cela, le choix de la clé « ligne » est très importante pour optimiser les requêtes sur la table.

Une colonne HBase est composée d'une « **column family** » et d'un « **column qualifier** », séparés par le caractère « **:** ».

Une **column family** permet de spécifier les colonnes d'une table et sont définies à la création de la table. Il est difficile de les modifier après la création. Chaque ligne contient les mêmes column family même s'il n'y a pas de valeur associée. Chaque column family a un set de propriétés de stockage qui permet de définir plusieurs paramètres, comme le cache en mémoire, la compression, etc.

Un column qualifier est ajouté au column family pour offrir un index pour une donnée. Contrairement aux column family, les column qualifier peuvent être totalement différents selon les lignes, et il peut y avoir de nombreux column qualifier pour chaque column family. Exemple : Avec un column family `content`, des column qualifier pourraient être `content:html` et `content:pdf`.

Une cellule est une combinaison d'une ligne, d'une column family et d'un column qualifier. Une cellule contient une valeur et un timestamp, qui représente la version de la valeur. Par défaut, le timestamp représente l'heure sur le RegionServer lorsque la donnée a été écrite. Il est possible de spécifier une autre valeur. Les données contenues dans une cellule n'ont aucun format particulier et sont stockées sous forme d'octets binaires.

Utilisation de HBase

Les actions possibles sur les données d'une table HBase utilisent l'interface Table et offrent les fonctions suivantes :

- `Result get(Get get)`
- `Result[] get(List<Get> gets)`
- `ResultScanner getScanner(Scan scan)`
- `ResultScanner getScanner(byte[] family)`
- `ResultScanner getScanner(byte[] family, byte[] qualifier)`
- `void put(Put put)`
- `void put(List<Put> puts)`
- `void delete(Delete delete)`
- `void delete(List<Delete> deletes)`

Il est possible de lancer le Shell HBase à l'aide de la commande

```
hbase shell
```

Et pour exécuter une classe Java :

```
yarn jar path/to/jar/file.jar
```

2.2.8. Apache Spark

Apache Spark est un Framework open source qui complémente Apache Hadoop afin de faciliter le développement d'applications traitant de grandes quantités de données. Il agit en tant que middleware entre Hadoop et les autres fonctionnalités offertes par les autres applications de l'écosystème Hadoop. Spark supporte les langages de développement R, Spark SQL, Java, Python et Scala.

Le principal avantage de Spark est qu'il effectue les tâches de manière plus rapide que le Map/Reduce traditionnel.

De plus, il offre une utilisation simple, rapide et contient de nombreuses librairies permettant d'effectuer tous types de traitements de données.

Il est compatible avec des sources de données HDFS, RDBMS (Système de gestion de base de données relationnelles), S3 (Simple Storage Service), Cassandra, Mongo DB.

Pour utiliser Apache Spark, il faut lancer la commande `spark-shell`.et dès lors vous pourrez exécuter des commandes Scala ou avec python en utilisant `pyspark`

Dès lors, il est possible d'exécuter des commandes dans le langage de son choix pour traiter les sets de données.

Pour exécuter un fichier Scala directement, il est possible d'utiliser l'argument `-i`. L'argument –master permet de définir quel sera le manager de ressource utilisé. L'argument `--num-executors` permet de définir le nombre d'exécuteurs seront chargés par Spark pour exécuter les tâches.

```
spark-shell --master yarn-client -i file.scala --num-executors 10
```

En particulier sur le cluster DAPLAB, il est nécessaire de spécifier le port, car il est partagé entre de nombreux utilisateurs. Pour ce faire, utiliser le paramètre suivant : `--conf spark.ui.port=$(shuf -i 2000-65000 -n 1)`.

2.2.9. Scala

Le langage Scala est l'un des langages utilisés par Spark et utilise une syntaxe proche du Java et du Python. Il utilise des Resilient Distributed Dataset (RDD), qui est une collection d'éléments distribués et immuables, pouvant être traitée en parallèle afin d'améliorer les performances.

Les fonctions basiques sur les RDD sont `map`, `persist`, `reduce` et `filter`.

Les variables sont définies à l'aide du mot clé `val`.

Lire des données

Pour lire les données depuis un fichier texte stocké dans HDFS, il suffit de le lire avec la commande `sc.textFile(filename)`. Dès lors, il est stocké dans un RDD où une ligne du fichier représente un élément du RDD.

Dans les cas particuliers où un fichier est stocké en plusieurs parties dans un dossier HDFS, comme c'est le cas avec Sqoop, il est possible de lire tout le contenu des fichiers dans le dossier avec la même commande : `sc.textFile(foldername)`.

Exécuter du SQL

Pour exécuter une requête SQL sur un set de données avec Spark, il est nécessaire d'importer le package suivant : `import sqlContext.createSchemaRDD`. Il est également nécessaire de préparer les colonnes de la table en spécifiant une classe pour spécifier les colonnes de la table : `case class Service(ip: String, port: String, software: String)`. Ainsi un exemple de requête SQL sur le set de données contenu dans le fichier `scan_services` peut être réalisé ainsi :

```
import sqlContext.createSchemaRDD

case class Service(ip: String, port: String,software: String)
```

```
val services = sc.textFile("scan_services_20150501").map(_.split(",", -
1)).map(row => Service(row(0), row(1), row(3)))
services.registerTempTable("services")

val results = sqlContext.sql("""select software, count(distinct ip) nb from
services where port='445' group by software order by nb desc""")
```

Passer des paramètres à un fichier

Pour passer un paramètre à un script Spark en ligne de commande, on peut utiliser la ligne suivante :

```
spark-shell -i < (echo val param = VALUE; cat <script-file-name>)
```

La variable `param` sera alors accessible dans le fichier scala comme une variable interne au script.

Écrire dans un fichier

Il est possible d'écrire dans un fichier du système local du serveur Hadoop avec la commande suivante :

```
scala.tools.nsc.io.File(filename).writeAll(stringValue)
```

Sauvegarder dans HDFS

Pour sauvegarder un RDD dans HDFS, il est possible d'utiliser la méthode `saveAsTextFile(filename)` du RDD. De cette manière, il est possible d'enregistrer le RDD sur le système de fichier HDFS. Il faut noter que cela produit un fichier HDFS par ligne du RDD. Pour éviter de séparer dans de nombreux fichiers, on peut utiliser la méthode `coalesce`, qui permet de définir le nombre de reducers utilisés et de forcer à un seul fichier de sortie sur HDFS.

2.2.10. Apache Sqoop

L'outil Sqoop de Apache offre la possibilité d'importer les données de sources SQL de manière automatisée dans l'architecture Hadoop. Il permet notamment d'importer dans le système de fichier HDFS directement, mais également dans des bases de données Hive ou Hbase.

Cet outil sera utilisé pour le transfert des données contenues sur les bases de données actuelles du NetObservatory.

Utilisation de Sqoop

Les commandes disponibles avec l'outil sqoop sont les suivantes :

- `codegen` Génère du code Java pour interagir avec la base de données
- `create-hive-table` Importe la définition d'une table dans Hive
- `eval` Exécute une requête SQL sur la base de données
- `export` Exporte un dossier HDFS dans une base de données
- `help` Liste les commandes et l'aide pour les commandes grâce à « `help command` »
- `import` Importe une table d'une base de données dans HDFS
- `import-all-tables` Importe toutes les tables d'une BDD dans HDFS
- `list-databases` Liste les bases de données sur le serveur
- `list-tables` Liste les tables pour la base de données sélectionnée
- `version` Affiche la version

Pour importer des données depuis une base de données MySQL dans Hadoop, il faut utiliser le programme sqoop avec les arguments suivants :

```
sqoop import --table nom_table --connect jdbc:mysql://host/db \
--username usr --password PWD
```

Il est tout à fait envisageable d'importer les données depuis d'autres types de bases de données que MySQL en utilisant un autre connecteur JDBC.

En ajoutant le paramètre `--hive-import`, Sqoop permet d'importer directement les données de MySQL dans Hive.

Dans ce cas, hive crée automatiquement les colonnes correspondantes dans Hive.

Il est également possible d'importer toutes les tables d'une base de données à l'aide de la commande `import-all-tables`.

On peut noter que Hive ne gère pas tous les types de données contenues dans les bases de données MySQL. Ainsi, les champs de types BLOB ne peuvent pas être importés dans Hive 0.14 avec sqoop 1.4.5. Néanmoins, HDFS accepte n'importe quel format de donnée et sqoop est capable d'importer les champs de types BLOB dans HDFS.

Il existe d'avantages de possibilités et d'options d'importations. La commande `sqoop help import` permet de les lister et d'en comprendre l'usage.

Le dossier de destination sur HDFS peut être spécifié à l'aide de l'argument `--target-dir`.

2.3. Conclusion

Cette analyse permet de mieux connaître les outils d'Hadoop, ainsi que l'infrastructure à disposition pour le projet NODaBOp et les interactions entre les scripts actuels du NetObservatory et les bases de données MySQL.

L'outil Sqoop sera utilisé pour importer les données du NetObservatory dans HDFS, dans un premier temps.

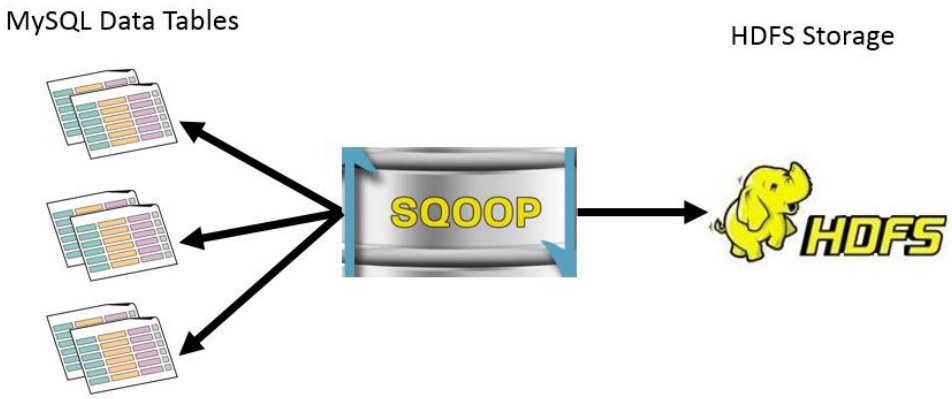


Figure 11 : Importation MySQL

Le tableau récapitulatif ci-dessous résume l'utilité de chacune des technologies présentées dans cette analyse.

Technologie	Utilité	Utilisé ?
HDFS	Stockage et redondance	
Map/Reduce	Opérations Map/Reduce	Intermédiaire généré par Spark
Hive	Requêtes SQL	Utilisé par Spark
Pig	Scripting	
HBase	Accès aléatoires et base de données orientée colonnes	Nécessite une remodélisation complète des données
Spark	Traitement des données Flexibilité, outil à tout faire	

Les technologies Hive et Pig permettant moins facilement de traiter des sets de données particuliers et comme les données des graphiques nécessitent divers traitement, c'est Spark qui a été choisi pour les opérations de traitement des données. HBase offre des performances accrues lorsqu'on souhaite effectuer des accès aléatoires sur les données, ce qui n'est pas le cas des scripts de statistiques du projet NetObservatory.

Figure 12 : Processing Spark

Spark a été choisi pour sa rapidité, sa simplicité et sa flexibilité. De plus, les requêtes des anciens scripts pourront partiellement être réutilisées dans Spark, car le langage Scala, utilisé par Spark permet d'effectuer des requêtes SQL.

3. Conception

3.1. Introduction

Le présent chapitre a pour but d'étudier les possibilités de mise à jour de l'architecture NetObservatory afin d'intégrer l'écosystème Hadoop dans le processus de récolte de données et de génération de graphiques.

3.2. Modélisation du processus

Le processus complet actuel se divise en cinq étapes, représentées dans la Figure 13.

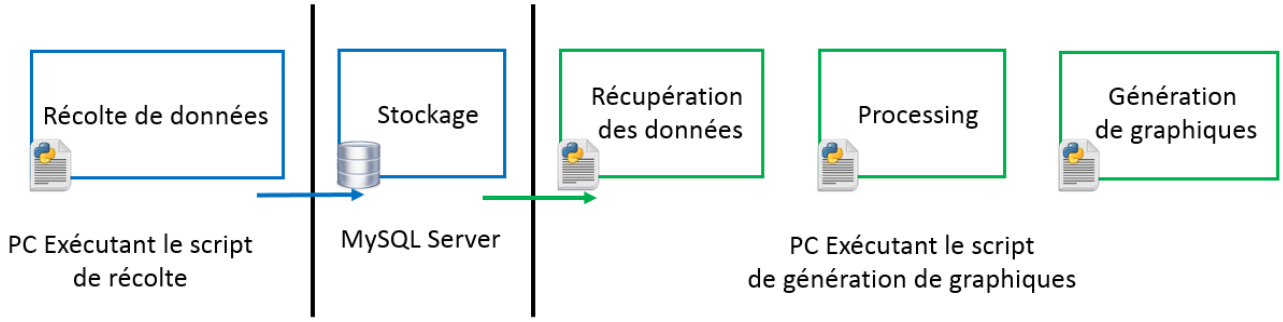


Figure 13 : Etapes MySQL

Ainsi, le premier script (en bleu) récolte les données et les stocke dans la base de données MySQL. Le second script (en vert) récupère les données dont il a besoin dans la base de données MySQL, et les utilise pour générer les données statistiques permettant de créer le graphique final.

Les sections suivantes présentent différentes approches pour intégrer l'écosystème Hadoop dans le processus du NetObservatory.

3.2.1. Méthode 1 : Uniquement stockage dans Hadoop

La première méthode possible pour intégrer Hadoop dans ce processus serait de simplement modifier le stockage, en effectuant une importation de la table MySQL générée dans l'écosystème Hadoop lorsque le script de récolte (bleu) a terminé son travail. Ainsi, les requêtes SQL exécutées par le script de génération de graphique (vert) seraient exécutées sur l'écosystème Hadoop plutôt que sur le serveur MySQL.

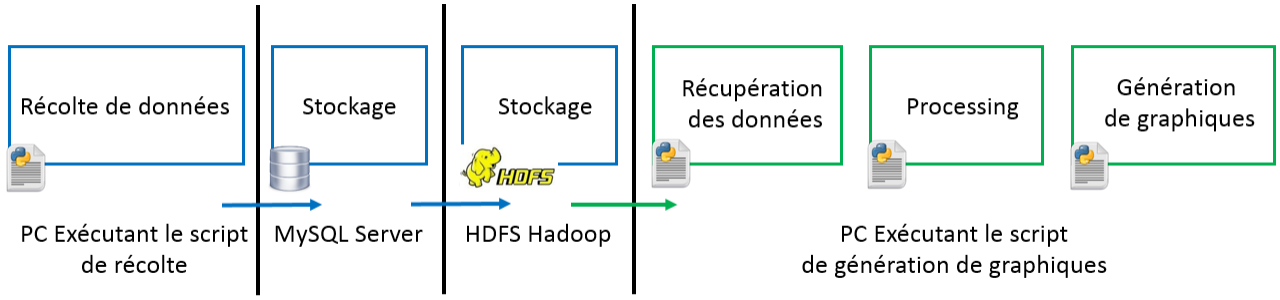


Figure 14 : Stockage dans HDFS

Cette méthode a pour seul changement de stocker les données dans HDFS plutôt que dans MySQL. Cette méthode n'offre aucun avantage, car les SGBD traditionnels offrent de meilleures performances qu'Hadoop pour ce type d'opérations sur de petits sets de données. Néanmoins, Hadoop étant prévu pour traiter de très grandes quantités de données, cette solution offre un stockage distribué et la scalability qui permettra à l'avenir de traiter d'avantages de données.

3.2.2. Méthode 2 : Processing Hadoop lors de la génération des graphiques

Les scripts de génération de graphiques utilisés actuellement utilisent les données reçues par le SGBD pour ne garder que certaines données, puis doivent parfois effectuer des actions sur le set de données reçu. Une amélioration possible de la méthode 1 serait d'intégrer ce travail dans l'infrastructure Hadoop afin d'avoir un traitement distribué sur les différentes machines du cluster.

Ainsi, dans la seconde méthode, le script de récolte (bleu) enregistre les données dans MySQL avant de les importer dans le système de fichier HDFS de Hadoop, comme dans la méthode 1. Lorsqu'on souhaite générer les graphiques, le script de génération de graphiques (vert) exécute un job sur le serveur Hadoop et celui-ci retourne uniquement les valeurs utiles au graphique. Ainsi, le traitement des données se fait de manière distribuée sur le cluster Hadoop.

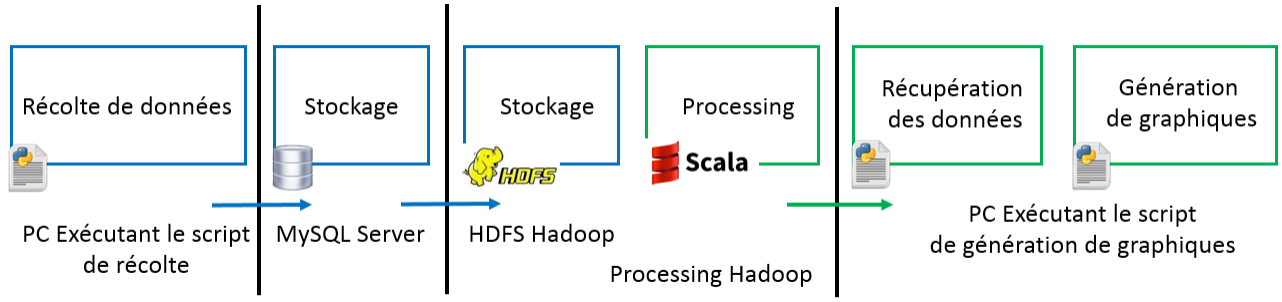


Figure 15 : Processing dans HDFS à la génération de graphiques

L'avantage de cette méthode est que le traitement des données est effectué par le cluster Hadoop, et non par le PC exécutant le script. De plus, les données résultant du Processing pourraient fournir une nouvelle source de Data Mining afin, par exemple, d'étudier l'évolution des statistiques dans le temps.

3.2.3. Méthode 3 : Processing Hadoop lors de la récolte

La troisième méthode consiste à déplacer l'exécution de la génération des données pour les graphiques directement à la fin du script de récoltes de données. En effet, comme ces dernières ne sont jamais modifiées entre temps, il serait tout à fait envisageable d'effectuer le travail de génération des données pour les graphique à la récolte. Ainsi, lorsqu'on souhaite générer les graphiques à l'aide des scripts de statistiques, le temps de traitement serait très faible.

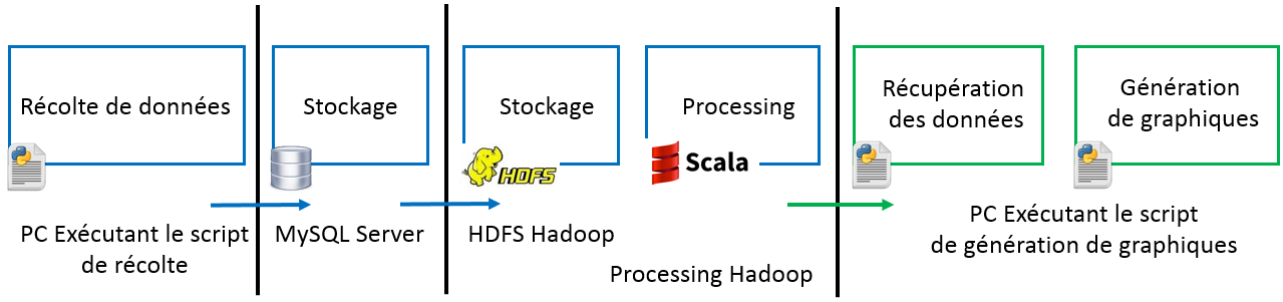


Figure 16 : Processing dans HDFS à la récolte

L'avantage de cette solution est que la phase de génération des graphiques est désormais presque instantanée, car les données sont prêtes à l'emploi.

Néanmoins, certains scripts de génération de graphiques nécessitant plusieurs tables, il sera nécessaire de réfléchir à l'ordre dans lequel seront exécutés les scripts et/ou effectuer un contrôle avant l'exécution des scripts de traitement des données.

3.2.4. Synthèse

Le tableau ci-dessous récapitule les avantages et inconvénients de chacune des méthodes possibles d'intégration de Hadoop dans le processus du projet NetObservatory.

Méthode	Avantages	Inconvénients
Méthode 1	Scalability	Traitement plus lent pour les gros sets de données, plus rapide pour les petits sets de données
Méthode 2	Scalability Traitement intégré à Hadoop Nouvelles sources de données	Pas de gain sur le temps de génération de graphiques
Méthode 3	Scalability Traitement intégré à Hadoop Nouvelles sources de données Génération de graphiques presque instantané	Nécessité de réflexion dans l'ordre d'exécution des scripts de récolte

Le tableau récapitulatif ci-dessus nous permet de voir tous les avantages proposés par la **troisième méthode**, qui consiste à intégrer le traitement des données et la génération des informations utiles pour la génération des graphiques à l'intérieur d'Hadoop. Le souhait du mandant étant d'améliorer le temps de génération des graphiques, c'est cette méthode qui semble la plus adaptée et qui, de ce fait, a été sélectionnée.

3.3. Importation des données dans Hadoop

L'écosystème Hadoop est prévu pour gérer de grandes quantités de données. Ainsi, de multiples opérations INSERT ne seraient pas adaptées pour l'insertion des données récoltées par les scripts Python, comme c'était le cas avec MySQL.

Deux choix principaux s'offrent alors à nous pour importer les données récoltées dans Hadoop :

- Conserver la base de données MySQL comme espace de transfert, et importer la table dans Hadoop une fois entièrement remplie.
- Enregistrer la totalité des données récoltées dans un gros fichier qui sera ensuite importé dans Hadoop

L'outil Sqoop de Hadoop offre la possibilité d'importer une ou plusieurs tables MySQL dans Hadoop de manière automatisée et parallèle, c'est-à-dire qu'il sépare le travail d'importation en quatre blocs (selon une clé primaire de la table).

De plus, cette méthode ne nécessite pas de changement dans les scripts de récoltes, et il est plus adapté de valider l'intégration de l'écosystème Hadoop dans le projet NetObservatory par étapes.

Le temps à disposition pour le projet étant limité, c'est la première méthode, qui consiste à conserver l'insertion dans MySQL et l'importation avec Sqoop, qui a été choisie pour l'importation des données.

3.4. Technologies utilisées

Le processus complet utilisé pour générer les données utiles pour les graphiques se fait en quatre étapes principales, comme illustré dans la Figure 17.

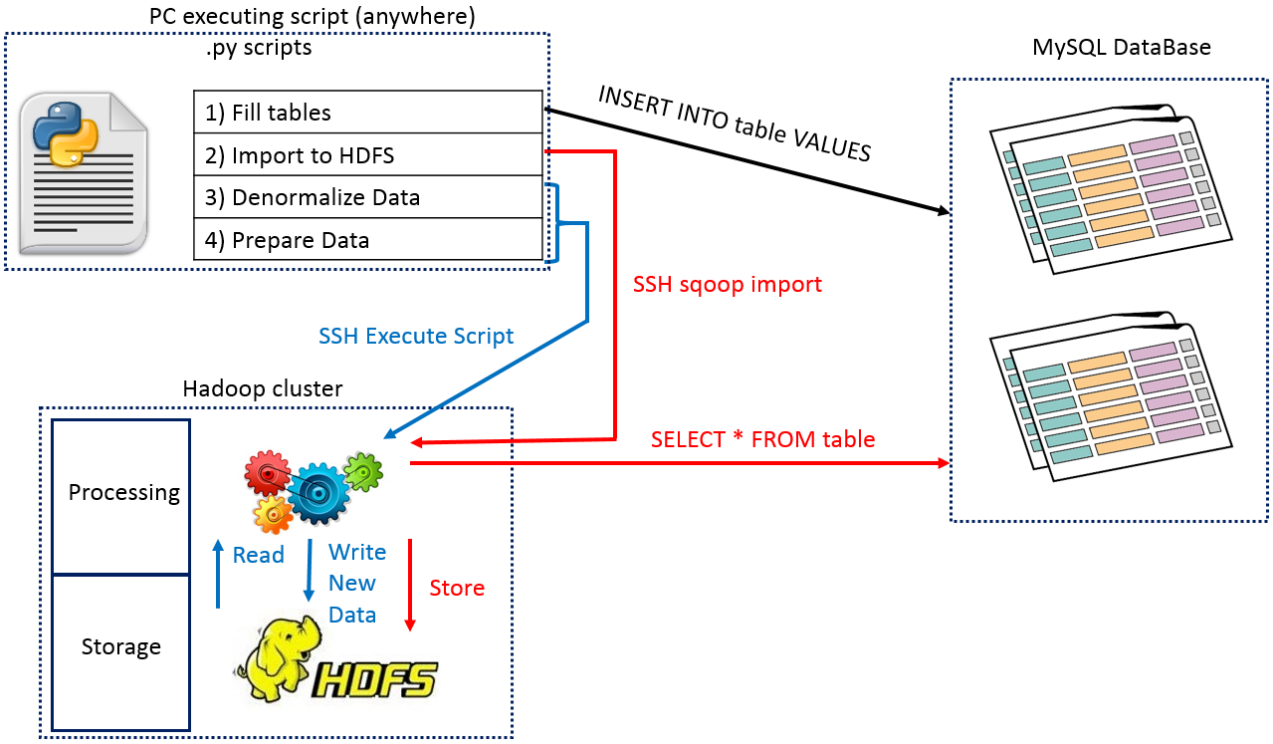


Figure 17 : Modélisation récolte

1. Récolte de données et remplissage des tables MySQL à l'aide des scripts actuels de récolte (pas de changement)
2. Importation de la nouvelle table dans HDFS à l'aide de l'outil sqoop, lancé au travers d'une commande SSH (en rouge sur la Figure 17).
3. Dénormalisation des données : Cette étape permet de dédupliquer certaines informations contenues dans d'autres tables afin de simplifier l'étape 4.
4. Utilisation des données pour générer les données qui seront utilisées par les scripts de génération de graphiques et enregistrement sur le système de fichier HDFS.

Les étapes 3 et 4 seront exécutées à l'aide de l'outil Spark de Hadoop. Spark supporte les langages Python, Java, Scala. Ainsi, les parties de traitement des données des scripts de génération de graphiques pourront simplement être déplacées sur le serveur Hadoop et exécutées par une commande SSH (en bleu sur la Figure 17). Le langage Scala étant un langage innovant qui prend de l'ampleur, c'est celui-ci qui a été choisi pour le développement des scripts de traitement des données dans Spark.

Une fois ces données sur le serveur HDFS, il suffira ensuite d'exécuter le script de génération de graphiques. Cette étape nécessitera beaucoup moins de temps et sera presque immédiate. En effet, les données seront déjà prêtes à être utilisées pour le graphique.

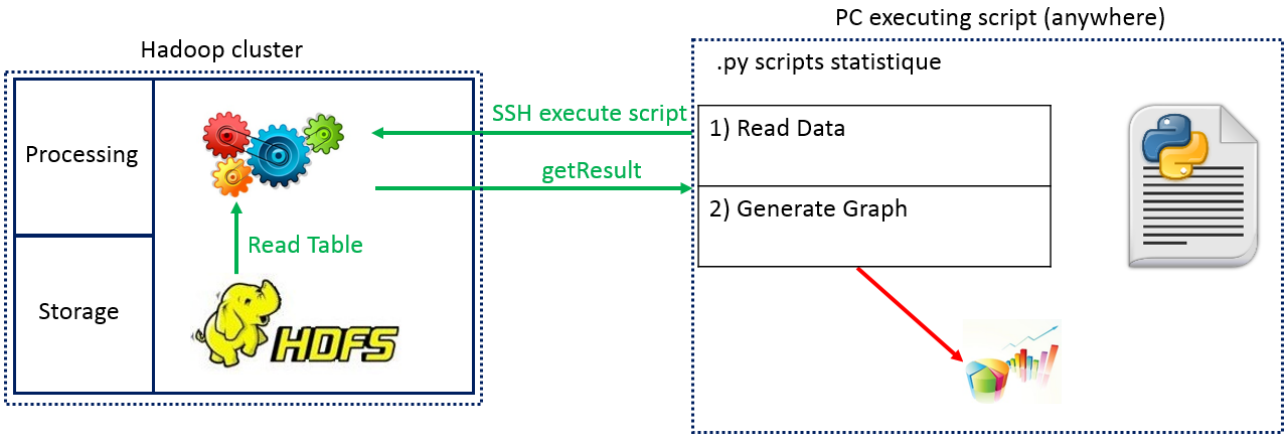


Figure 18 : Modélisation statistique

Le script python de génération de graphiques effectuera une simple commande SCP pour récupérer le fichier de résultat puis générera le graphique à partir des données récupérées.

3.5. Modélisation des données

Comme défini dans le chapitre 3.4, certaines tables nécessitent une étape de dénormalisation des données, afin de rendre le traitement des données plus rapide. La modélisation des données est différente pour chacune des tables et nécessite une étude approfondie des scripts de génération de graphique. En effet, il est nécessaire de connaître les données nécessaires pour générer le graphique afin d'optimiser la requête pour récupérer ces données.

3.5.1. Dénormaliser

La dénormalisation des données permet d'effectuer des requêtes de manière plus rapide et efficace. Cette étape consiste à dupliquer et réduire, ou augmenter certaines informations afin de ne traiter que celles qui nous intéressent.

Par exemple, dénormaliser le nom d'un logiciel avec sa version permet d'effectuer des requêtes simplifiées sur un champ « software » plutôt que sur la version complète.

Ainsi, le champ « Software » avec des valeurs pour plusieurs lignes « Wordpress 3.4.5 », « Edition speciale Wordpress 2.4.5.1 » et « Joomla 4.2 » serait dupliqué dans la colonne short_software avec les valeurs « wordpress », « wordpress » et « joomla ». Ainsi, une requête ne concernant que les logiciels « wordpress » pourrait être exécutée très rapidement.

Dans les cas spécifiques des scripts sélectionnés, aucune dénormalisation des données n'a été nécessaire.

3.6. Confidentialité des données

Les données sauvegardées par les scripts de récoltes étant considérées comme confidentielles, et le serveur DAPLAB étant partagé entre plusieurs utilisateurs, il sera nécessaire de s'assurer que les données ne pourront pas être accessibles par les autres utilisateurs.

Les permissions pourront être définies comme sur un système Linux traditionnel sur chacun des fichiers importés à l'aide de la commande :

```
hadoop fs –chmod 700 file
```

3.7. Conclusion

Le chapitre de conception énumère les différentes possibilités d'intégration de l'architecture existante dans un contexte Hadoop. Le prochain chapitre détaillera sa mise en place.

4. Implémentation

4.1. Introduction

Le chapitre d'implémentation a pour but de détailler les étapes effectuées pour respecter les points définis par les objectifs.

4.2. Architecture de l'infrastructure

Dans le cadre de ce projet, nous utiliserons plusieurs éléments du réseau de l'HEIA-FR, répartis sur plusieurs sous-réseaux. Le schéma de la Figure 19 décrit l'architecture qui a été utilisée dans ce projet.

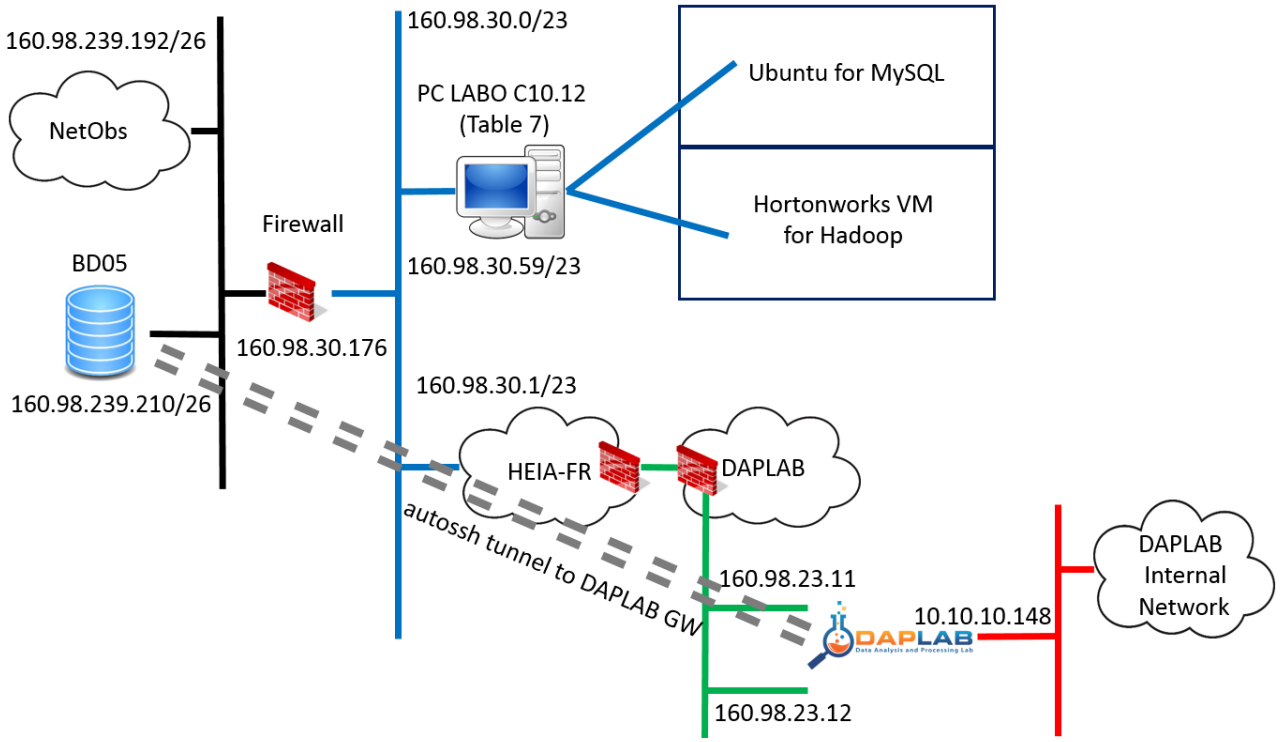


Figure 19 : Infrastructure existante

4.2.1. Environnement de test local

Le PC du laboratoire est l'hôte de deux machines virtuelles :

- Ubuntu client utilisée en tant que serveur MySQL et gateway pour accéder à tous les différents éléments et exécuter les scripts
- Hortonworks Virtual Machine contenant une Sandbox contenant tous les outils Hadoop pour les premiers essais

4.2.2. Architecture finale

Dans la nouvelle architecture souhaitée, les données seront stockées sur le cluster DAPLAB, un cluster Hadoop mis en place à la HEIA-FR et utilisé pour divers projets.

Pour accéder aux serveurs de base de données du NetObservatory directement depuis le réseau interne de l'école, une entrée NAT a été ajoutée sur le Firewall entre les réseaux NetObs et HEIA-FR. Le serveur MySQL sera accessible depuis ce réseau sur l'adresse `160.98.30.240`. Pour que le serveur MySQL puisse ouvrir une connexion SSH vers le DAPLAB, il est encore nécessaire de spécifier une route statique vers le DAPLAB :

```
add route 160.98.23.0 255.255.255.0 160.98.239.220
```

Pour traverser les nombreux Firewalls jusqu'au serveur MySQL, un tunnel reverse SSH est ouvert depuis le serveur MySQL NetObservatory pour que les données MySQL puissent être accessibles directement depuis la Gateway du DAPLAB.

Ce tunnel est effectué à l'aide de l'outil `autossh`.

```
autossh -R 10.10.10.148:33306:localhost:3306 netobservatory@pubgw1.daplab.ch
```

Pour fonctionner, la clé publique SSH de l'utilisateur doit être activée sur le DAPLAB (Contact : Benoit Perroud) et la clé privée correspondante doit être installée sur le serveur MySQL NetObservatory.

L'outil `autossh` offre une communication permanente en ssh entre la machine sur laquelle on exécute la commande et la cible (ici `netobs@pubgw1.daplab.ch`). Le paramètre `-R 10.10.10.148:33306:localhost:3306` permet de spécifier un port forwarding sur la machine de destination vers la machine source, au travers du tunnel SSH.

Cette commande permet à la Gateway du cluster d'accéder au serveur MySQL, mais les différentes machines faisant partie du cluster n'y auront pas accès. Ainsi, un port-forwarding a été mis en place sur la Gateway pour retransmettre les paquets arrivant sur l'adresse `10.10.10.148:33306` (sa connexion vers les autres machines du cluster) sur le port 33306 local, et ainsi atteindre le serveur MySQL.

Dès lors, le serveur MySQL sera accessible à partir de l'adresse IP `10.10.10.148` et le port `33306` depuis la Gateway du cluster.

4.3. Importation des tables MySQL

Deux méthodes ont été utilisées pour importer les tables MySQL sur le DAPLAB, la première étant l'outil Sqoop et la seconde une importation manuelle depuis un fichier SQL.

4.3.1. Importation avec Sqoop

Pour importer les tables dans HDFS à l'aide de l'outil Sqoop, il y a deux choix : l'import de toutes les tables ou l'import d'une seule table.

L'import de toutes les tables d'une base de données d'un seul coup est soumis à plusieurs conditions :

- Les tables ne doivent pas déjà exister dans la destination
- Les tables doivent avoir un index (clé primaire) sur une seule colonne afin de permettre la parallélisations de l'import (ou le paramètre `--autoreset-to-one-mapper` doit être défini)
- On souhaite importer toutes les colonnes de toutes les tables

La commande suivante doit être exécutée sur le cluster Hadoop :

```
sqoop import --table tbl_name_date --connect jdbc:mysql://10.10.10.148/archive
    --username usr --password pwd --target-dir netobs/tbl_name/tbl_name_date
```

Les tables suivantes doivent également ajouter le paramètre `–m 1` car elles n'ont pas d'index.

bgpresults	scan_os
nvd_cvss	scan_services
nvd_vulns	web_fingerprint_full
reverse_dns	whois

4.3.2. Importation manuelle

Suite à des problèmes de réseau ne permettant pas d'importer les données sur le DAPLAB depuis le serveur MySQL, ainsi que des problèmes du serveur MySQL BD05 lui-même, certains fichiers ont été importés manuellement à partir des backups mysqldump, accessibles depuis le serveur BD05.

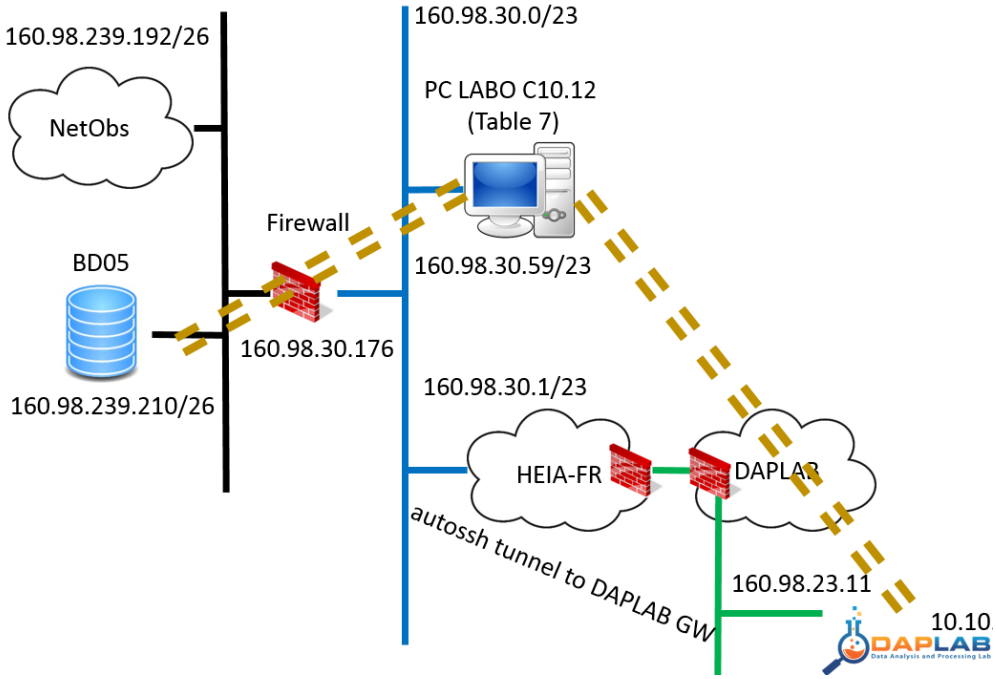


Figure 20 : Importation manuelle

L'importation s'est faite depuis le PC Labo, avec tout d'abord une copie par SCP sur la machine locale et une seconde copie par SCP vers le DAPLAB.

Les fichiers SQL ont été traduits en CSV à l'aide d'un script Python développé par James Mishra[1] à l'aide de la commande suivante :

```
zcat backup_file.gz | ./mysqldump_to_csv.py > file.csv
```

Puis importées dans le système HDFS à l'aide de la commande :

```
hadoop fs –copyFromLocal file.csv netobs/table_name/table_name_date.csv
```

4.3.3. Importation de plusieurs tables

Pour importer une grande quantité de tables sur le DAPLAB, il existe deux méthodes, selon si l'on préfère effectuer le travail à l'aide de l'outil Sqoop ou manuellement avec SCP à partir des dumps MySQL.

Les deux méthodes peuvent être réalisées avec l'utilitaire `bulk_import.py` disponible sur le CD.

L'utilitaire `bulk_import.py` prend deux paramètres, le premier étant le type de requêtes et le deuxième le nom du fichier contenant les tables à importer, une table par ligne, sans espace. Des informations détaillée de l'utilisation de l'utilitaire sont disponibles à l'aide de la commande « `bulk_import.py help` ».

[1] Source du script :https://github.com/jamesmishra/mysqldump-to-csv

Sqoop

Tout d'abord, il faut vérifier que le tunnel SSH entre le serveur de base de données et le cluster Hadoop est ouvert (Voir chapitre 4.2 Architecture ci-dessus)

Le nom des tables dans le fichier `table_filename` doit être dans le format suivant : `tablename_yyyymmdd`.

Sur le DAPLAB, exécuter le script `./bulk_import.py sqoop table_filename`. Il sera peut-être nécessaire de modifier la configuration du connecteur de la base de données dans le script Python.

SCP

Le nom des tables dans le fichier `table_filename` doit être dans le format suivant : `tablename_yyyymmdd.gz`.

Tout d'abord, copier tous les fichiers mysqldump sur le DAPLAB à l'aide de la commande SCP. Note : L'espace disque à disposition sur le DAPLAB est limité

Puis, sur le DAPLAB, exécuter le script suivant : `./bulk_import.py hadoop table_filename`.

4.4. Organisation des fichiers

Afin de faciliter la compréhension de l'organisation des différents fichiers du projet, les sections suivantes résument les différents fichiers et dossiers créés dans le cadre de ce projet.

4.4.1. DAPLAB

Les fichiers sont organisés ainsi sur le système de fichier local du DAPLAB, la racine étant `/home/netobservatory`.

Chemin	Description
`/scripts`	Contient tous les scripts Scala
`/results`	Contient les résultats sous format texte, un dossier par graphique et un fichier par date. La dernière version du fichier est nommée nom_graph_last.
`/csvdata`	Fichiers sous format CSV pour un transfert manuel
`/`	Fichiers Java auto-générés pour l'importation

Les fichiers sur le système de fichiers HDFS du DAPLAB sont organisés ainsi, la racine relative sur HDFS étant `/user/netobservatory`.

Chemin	Description
`/netobs/`	Contient tous les fichiers et dossiers du NetObservatory
`/netobs/table_name`	Un dossier par table est créé
`/netobs/table_name/table_name_date`	Un dossier par date est créé

4.4.2. Gateway NetObservatory

Une machine virtuelle Ubuntu a été installée pour servir de machine « Gateway ». Les scripts Python de récolte et de génération de graphiques peuvent y être exécutés. Les fichiers liés au NetObservatory sont organisés ainsi, la racine étant `/home/netobs/netobs`.

Chemin	Description
`/`	Contient tous les fichiers relatifs au NetObservatory
`/scripts`	Contient les scripts de récolte
`/scripts/report`	Contient les scripts de génération de graphiques
`/netobs.conf`	Fichier de configuration de la base de données

4.5. Choix des scripts statistiques à implémenter

Il a été nécessaire d'effectuer un choix sur les différents scripts de statistiques à implémenter. Afin de valider la faisabilité des différents graphiques du rapport NORA et de pouvoir comparer les performances des deux systèmes, il a été décidé de sélectionner trois graphiques différents :

- Un graphique nécessitant une simple table (graphique en barre)
- Un graphique utilisant plusieurs tables sur plusieurs années (graphique histogramme)
- Un graphique travaillant sur une grande quantité de données

Les graphiques présentés ci-dessous sont tirés du rapport NORA Q2[2], généré en juin 2013.

4.5.1. Graphique sur une table

Le premier graphique choisi comporte quatre résultats représentés simplement en barres.

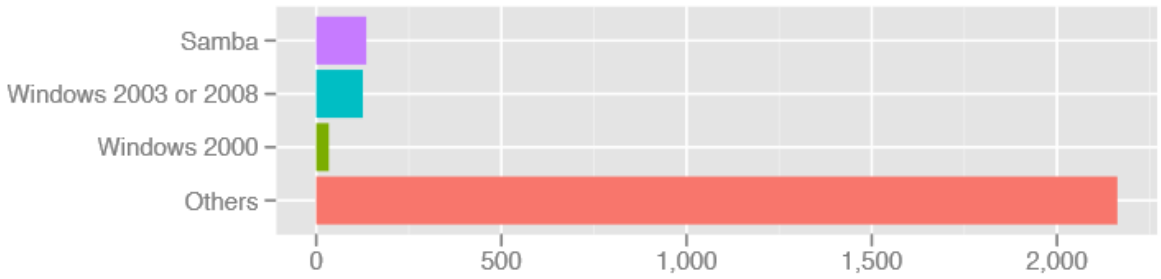


Figure 21 : Graphique barre msds_stats NORA [7]

Ce graphique a pour but de recenser le nombre de serveurs ayant le port 445 (Samba) ouvert et accessible depuis l'extérieur.

[2] NORA Report : https://www.netobservatory.ch/static/static_files/nora/archives/2013-Q2_NORA.pdf

4.5.2. Graphique histogramme

Le second graphique choisi comporte trois résultats empilés dans un graphique histogramme sur une année.

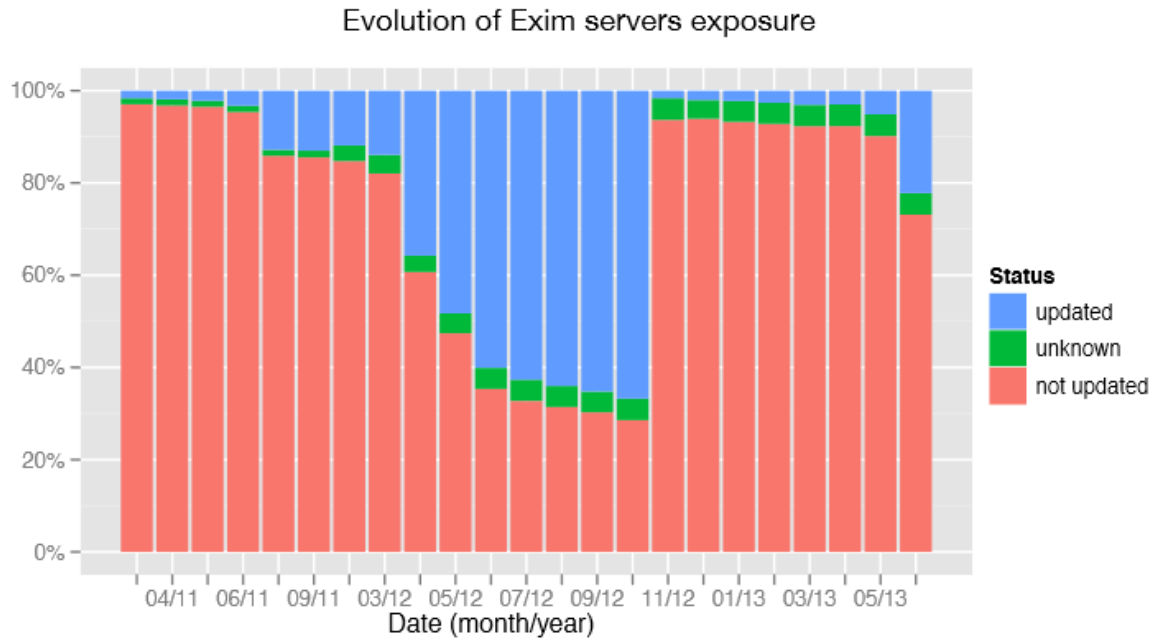


Figure 22 : Graphique histogramme NORA 4013_build_smtp_vuln

Il permet de déterminer si les serveurs Mail de type exim et sendmail sont à jour par rapport aux vulnérabilités trouvées.

4.5.3. Graphique sur une grosse quantité de données

Pour ce dernier cas, comme aucun des graphiques existants ne convenait, il a été décidé de développer un nouveau script traversant la table la plus grosse de notre base de données, à savoir la table webfingerprint_full. Dès lors, il fallait trouver une information pertinente à récupérer dans la table.

Ainsi, il a été décidé de créer un graphique recensant les dix DOCTYPE HTML les plus utilisés sur le Web suisse.

4.5.4. Génération des graphiques

Suite à des problèmes de librairies lors de l'installation du programme R, utilisé par les anciens scripts pour la génération de graphiques, les scripts ont été générés à l'aide de l'outil GNUPlot, depuis la librairie *gnuplot-py*. Des exemples d'utilisation de gnuplot sont disponibles sur le CD.

4.6. Passage du nom des tables aux Scripts Scala (Hadoop)

Les scripts Scala ont été développés de manière à ce que la date puisse être passée en paramètre au script Scala. Ces paramètres devront être passés directement par le script de récolte, ce dernier ayant accès aux noms des tables MySQL des derniers mois à l'aide des fonctions développées dans la librairie `db`.

4.6.1. Une table

Lorsque le script ne traite les données que dans une table, le nom de la table est enregistrée dans la variable `date`, sous forme de chaine de caractères pour un script n'utilisant qu'une seule date.

```
val date = "nom_table_date"
```

4.6.2. Plusieurs tables

Dans le cas où il s'agit d'un seul script utilisant plusieurs tables, il enregistre les paramètres dans la variable `table_list` la variable table_list sera dans le format suivant :

```
val table_list = List("nom_table_date", "nom_table2_date")
```

4.6.3. Histogramme

Pour un histogramme la variable `table_list` est une liste de liste qui doit être dans le format suivant :

```
val table_list = List(List("mm/yy", "nom_table_date1"), List("mm/yy", "nom_table_date2"), …)
```

4.7. Stockage des résultats

Les résultats sont stockés chacun dans un fichier texte, avec la date de la production du graphique sous le format `nom_script_statistique_yyyymmdd`.

Le dernier fichier de statistique est disponible sous `nom_script_statistique_last`.

4.8. Accès au DAPLAB

Pour accéder au DAPLAB, le cluster Hadoop de la HEIA-FR, il a été nécessaire de générer une paire de clé privée/publique. En effet, la connexion se fait uniquement par clé et non par mot de passe. M. Benoit Perroud est responsable des accès au DAPLAB.

Les points suivants résument les étapes utilisées pour simplifier la création, l'installation et l'utilisation d'un clé SSH. L'utilisateur netobservatory a été créé spécifiquement pour la continuïté du projet

4.8.1. Création de la clé

L'utilitaire puttygen[3] permet de générer une clé privée avec une interface graphique intuitive.

[3] PuttyGen : http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

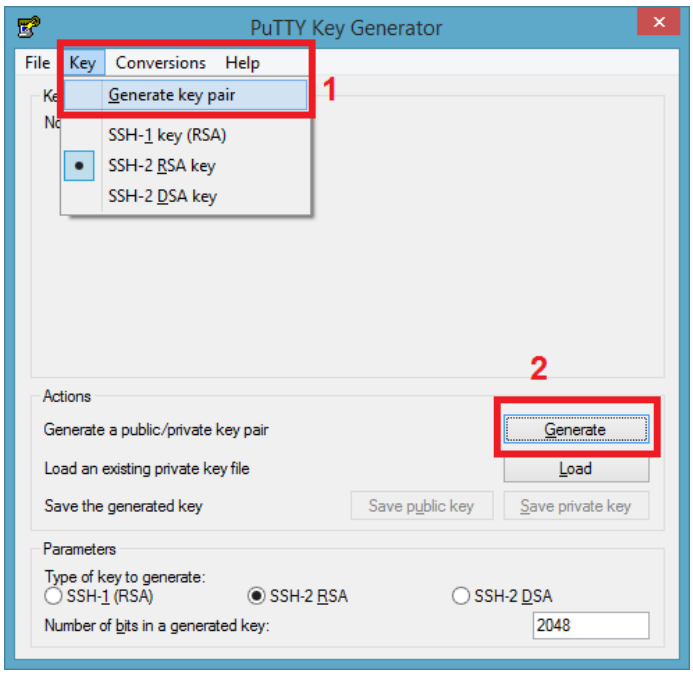


Figure 23 : Génération de clé avec PuttyGen

Cliquez sur Generate pour générer une clé privée, entrez une « Key passphrase » puis enregistrez la paire de clé privée / publique à l'aide des boutons « Save private key » respectivement « Save public key ». La clé publique doit être envoyée à Benoit Perroud pour l'installation sur le DAPLAB et la clé privée doit être conservée.

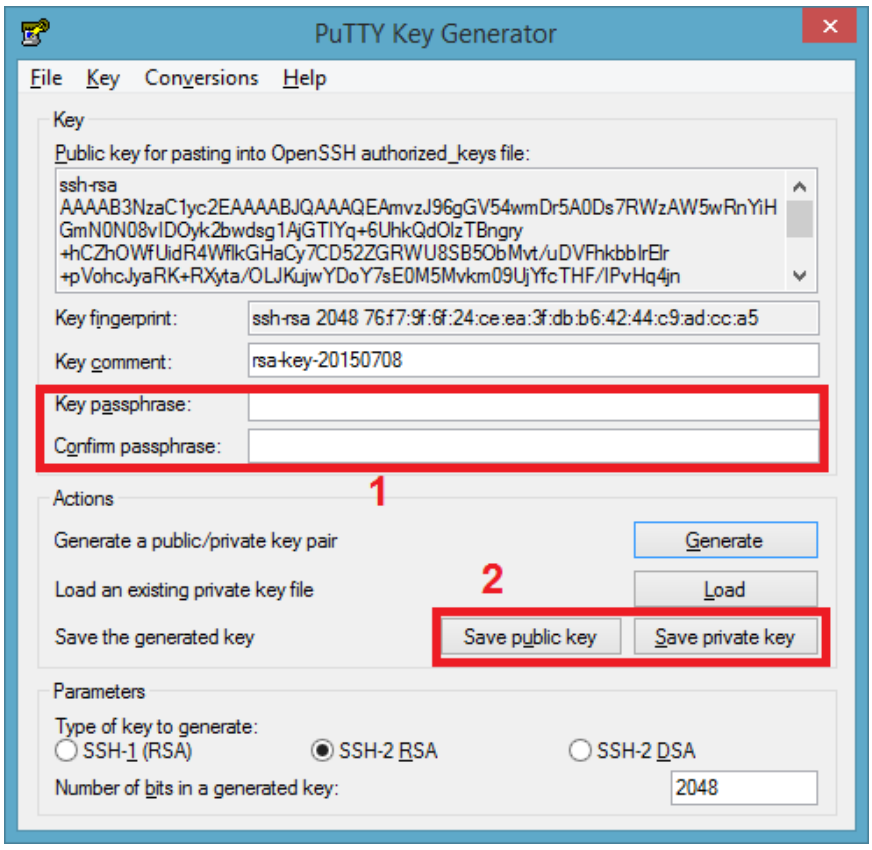


Figure 24 : Sauvegarder la clé

Pour pouvoir l'utiliser sur un environnement Linux, il est nécessaire de la convertir en clé openssh.

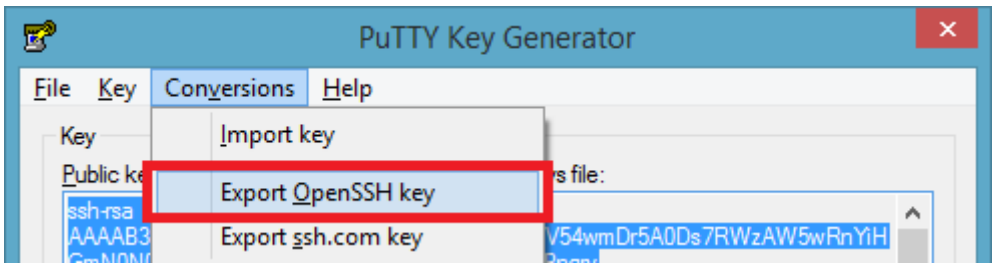


Figure 25 : Exportation de la clé pour Linux

4.8.2. Installation sur Linux

Pour installer la clé ssh sur Linux et simplifier son utilisation, il est conseillé de créer un fichier de configuration pour vos serveurs.

La clé doit être enregistrée avec les permissions `chmod 700`, dans le dossier `~/.ssh` de votre utilisateur.

Il est dès lors possible de configurer un fichier de configuration. Pour ce faire, il suffit d'ouvrir le fichier `~.ssh/config` avec votre éditeur préféré et d'y ajouter les lignes suivantes :

```
Host daplab
    HostName pubgw1.daplab.ch
    IdentityFile ~/.ssh/private_key
    User netobservatory
```

4.8.3. Utilisation sur Linux

Maintenant que votre clé est configurée, vous pouvez l'utiliser très facilement à l'aide de la commande `ssh daplab` (valeur définie dans "Host")

4.8.4. Utilisation sur Windows (Putty)

Pour utiliser la clé SSH dans Putty, il est nécessaire de l'ajouter dans l'utilitaire sous « Connection » > « SSH » > « Auth ».

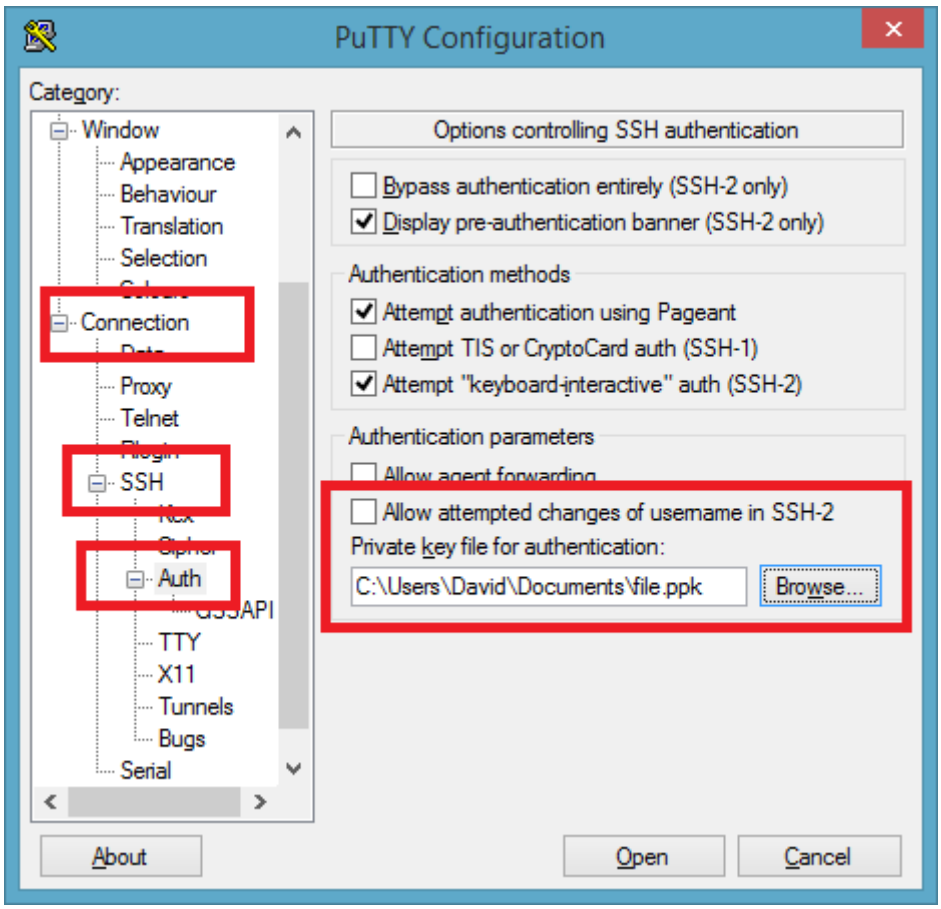


Figure 26 : Utilisation avec Putty sur Windows

Puis aller dans Session et entrez netobservatory@pubgw1.daplab.ch dans le champ Host Name.

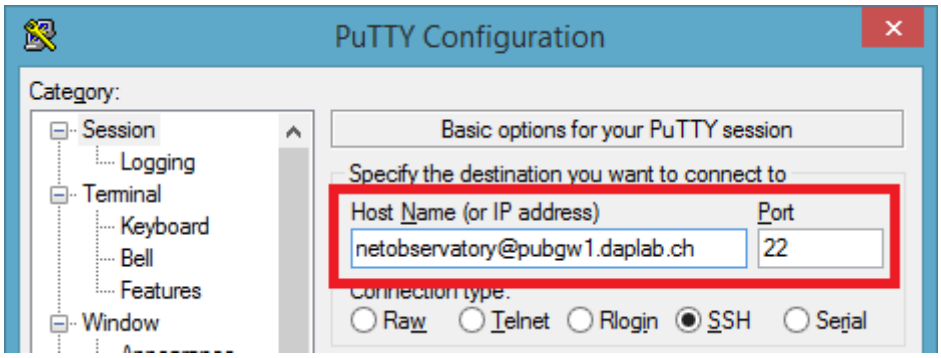


Figure 27 : Démarrer la connexion SSH

L'accès au DAPLAB se fait avec votre clé privée. Il est conseillé d'enregistrer la configuration pour éviter de devoir effectuer cette manipulation à chaque connexion.

4.9. Conclusion

Le chapitre d'implémentation permet de revenir sur les différents problèmes rencontrés dans ce projet et permet de faciliter la reprise du projet à l'avenir. Le chapitre de tests et validation permettra de valider les fonctionnalités implémentées et de comparer les performances entre les deux solutions.

5. Tests et validation

5.1. Introduction

Ce chapitre permet dans un premier temps de valider le bon fonctionnement des différentes étapes développées lors de ce projet, puis d'effectuer des tests de performance afin de comparer les résultats entre les deux solutions.

5.2. Tests fonctionnels

Les différents tests fonctionnels pour valider le bon fonctionnement des différents outils utilisés et scripts développés ont été effectués tout au long de l'implémentation, par étapes. Les résultats des différents tests sont définis dans l'annexe 8 de ce document.

5.2.1. Importation des données

Les tests d'importation des données ont été effectués avec l'outil Sqoop depuis le serveur local Hadoop d'abord, avec une version de chacune des tables. Le résultat importé a pu être validé avec plusieurs requêtes sur chacune des tables.

Les mêmes tests ont été effectués sur le serveur DAPLAB par la suite pour valider la bonne marche des importations.

Suite au problème de connexion entre le serveur MySQL et le cluster Hadoop, l'importation s'est fait manuellement pour les tables `smtp_discover` et `nvd_vulns`, utilisées pour la génération du graphique `4013_build_smtp_vuln`. Le contenu des nouvelles tables a pu être validé de la même manière.

5.2.2. Processing dans Hadoop

Dans le cadre de l'implémentation en Scala, les résultats retournés ont été comparés à ceux reçus sur le serveur MySQL en place. Ainsi, il a été possible de valider le bon fonctionnement du script.

Pour implémenter un script Scala, il est conseillé de tout d'abord coder le script dans l'invite de commande `spark-shell` étape par étape. Une fois le fonctionnement du script validé, on peut le sauvegarder dans un fichier `.scala` et le lancer avec la commande `spark-shell –i file.scala`. De plus, il est conseillé d'effectuer les tests sur un petit échantillon de données avant de s'attaquer à un set complet. Ainsi, on peut vérifier que le fonctionnement voulu est bien respecté et contrôler visuellement si c'est le cas.

5.2.3. Communication Gateway – Hadoop

Il est prévu de pouvoir exécuter les scripts utilisés par NetObservatory depuis une machine annexe que l'on nommera ici Gateway.

Pour effectuer des opérations sur Hadoop à distance, il est nécessaire de s'y connecter en SSH (Secure SHell). Dans le cas du Hadoop local, il suffisait d'une combinaison nom d'utilisateur et mot de passe. Dans le cas de l'accès au cluster DAPLAB de l'école, il est nécessaire de se connecter par l'intermédiaire d'une clé SSH.

Les deux méthodes ont été testées, tout d'abord en exécutant une simple commande `ls` et en récupérant le résultat, puis en exécutant une commande `spark-shell` lançant un script Scala.

5.2.4. Génération des graphiques

Les graphiques produits par les scripts correspondent à ceux générés par les anciens scripts de génération de graphiques.

Script 6012_msds_build

Le service Samba, fonctionnant sur le port TCP 445, est connu pour ses problèmes de sécurité, notamment sur le système d'exploitation Windows. Des milliers des virus et de vers utilisent ce port pour infecter d'autres ordinateurs sur Internet. Des virus et des vers ont probablement infectée une bonne partie des serveurs montrés dans ce graphique. L'ouverture de ce protocole au public est un énorme risque et devrait être évité.

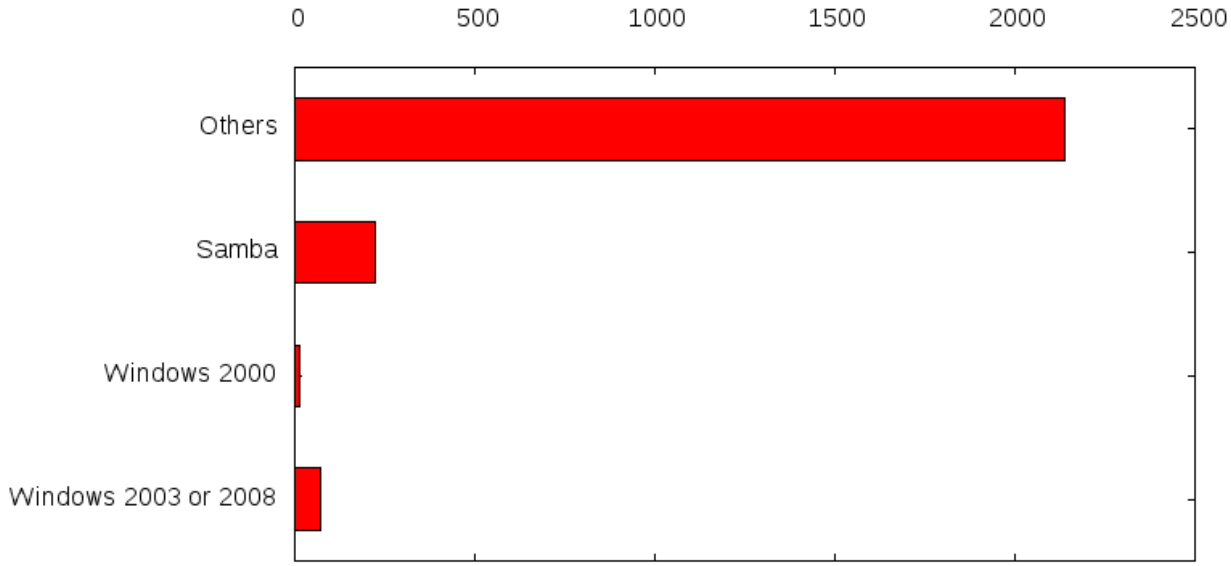


Figure 28 : Script 6012_msds_build graphique

Comme on peut le voir dans la Figure 28, le nombre de serveurs Windows 2000, 2003 et 2008 visibles depuis l'Internet a baissé depuis le rapport NORA de 2013. Néanmoins, le nombre global de serveurs ayant toujours un port TCP 445 ouvert reste tout de même important.

Script 4013_build_smtp_vuln

Ce graphique permet de déterminer si les systèmes utilisant l'application Exim ont des vulnérabilités connues. Comme on peut le voir dans la Figure 29 ci-dessous, les données des derniers mois indiquent qu'une faille importante a été découverte en octobre 2014.

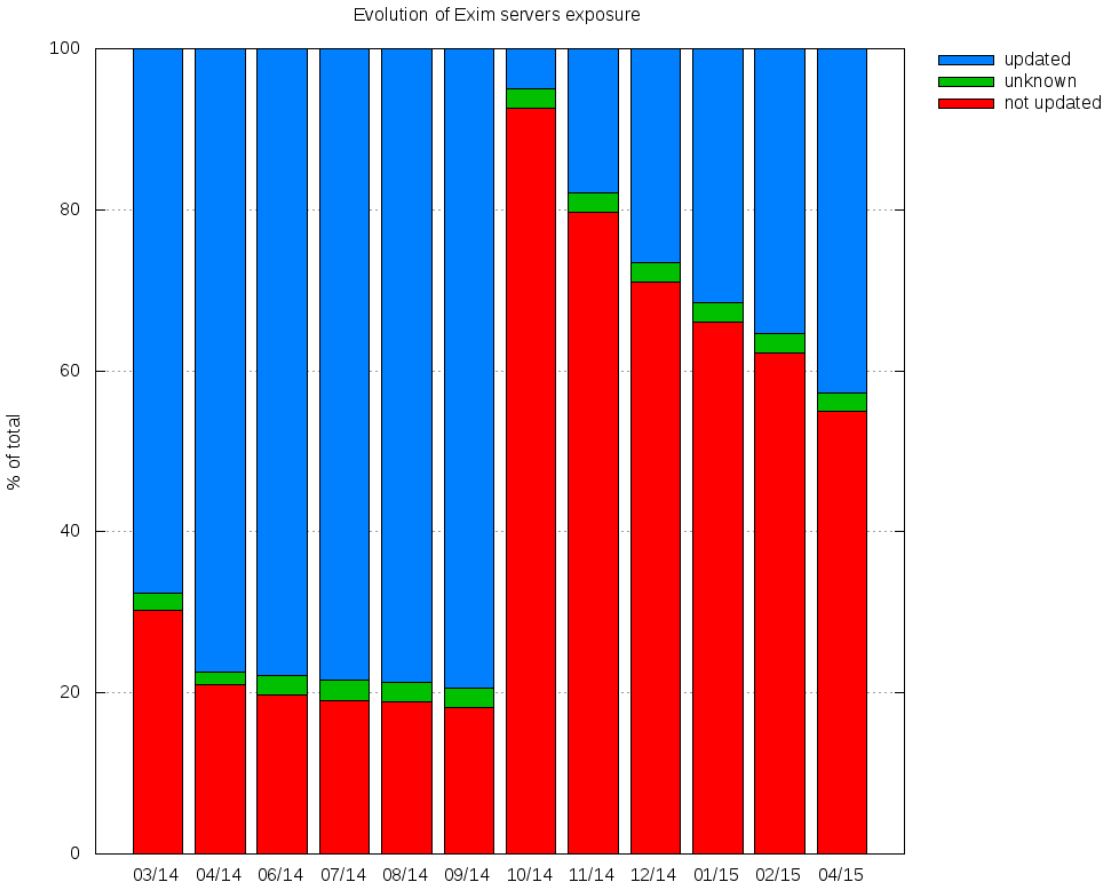


Figure 29 : Script 4013_build_smtp_vuln graphique

Cela coïncide avec la découverte d'une faille dans openssl[4] ou à la vulnérabilité ghost[5] qui concerne également les serveurs exim. Le graphique est donc cohérent avec la réalité.

[4] OpenSSL : http://blog.syloe.com/tag/openssl/

[5] Ghost : https://fr.wikipedia.org/wiki/Ghost_%28Vuln%C3%A9rabilit%C3%A9%29

5.3. Tests de performance

Les tests de performance permettent de déterminer quelle architecture est la plus performante entre l'ancienne architecture sur MySQL et la nouvelle sur Hadoop.

5.3.1. Méthodologie

Afin de déterminer la meilleure architecture, un test de performance sera effectué sur chacune des étapes du processus de génération de graphiques.

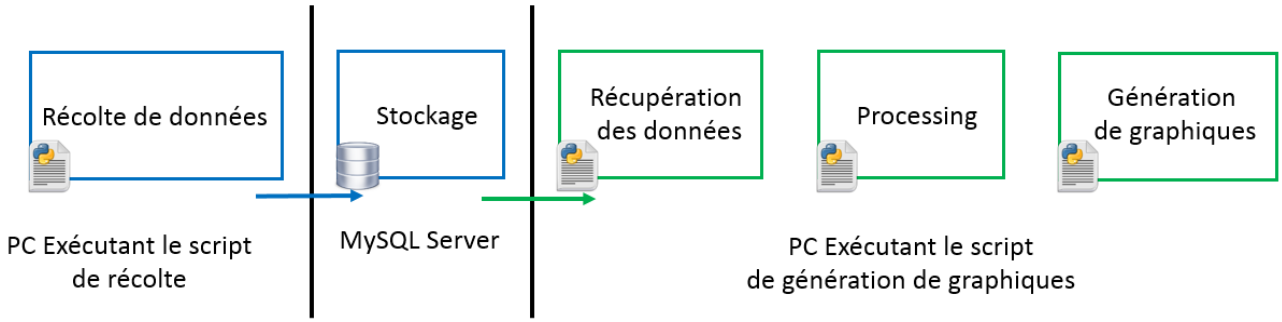


Figure 30 : Processus ancienne architecture

La première partie de récolte et d'insertion des données dans MySQL étant la même, elle ne sera pas évaluée. Sur l'architecture MySQL, les temps d'exécution de la récupération de données, de la mise en forme des données et de la génération des graphiques seront calculés.

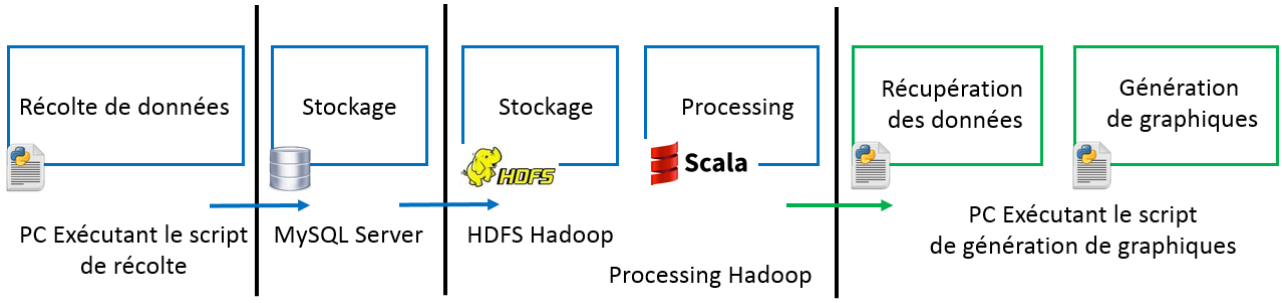


Figure 31 : Processus nouvelle architecture

Sur la nouvelle architecture Hadoop, les temps d'exécution de l'importation (Sqoop), du traitement des données, de la récupération des données entre Hadoop et la Gateway et de génération de graphique seront calculés.

5.3.2. Résultats

Les résultats des tests de performances, détaillés dans l'annexe 9 ont démontré que les temps de traitement des données des graphiques sont plus courts sur l'ancienne architecture pour des sets de données très petits (20MB). Cependant, les performances pour des sets de données plus grands sont plus rapides sur l'architecture Hadoop.

Le tableau récapitulatif et le graphique ci-dessous résument les résultats obtenus pour les tests de performances sur chacune des architectures :

Tableau récapitulatif				
	Data[MB]	**MySQL**	**DAPLAB**	**Remarques**
Script 1	22,71	7,91	29,40	1 Fois petite quantité de données
Script 2	1219,8	48,71	304,066	24 Fois petites quantités de données
Script 3	3000	138,27	108,33	1 Fois grande quantité de données

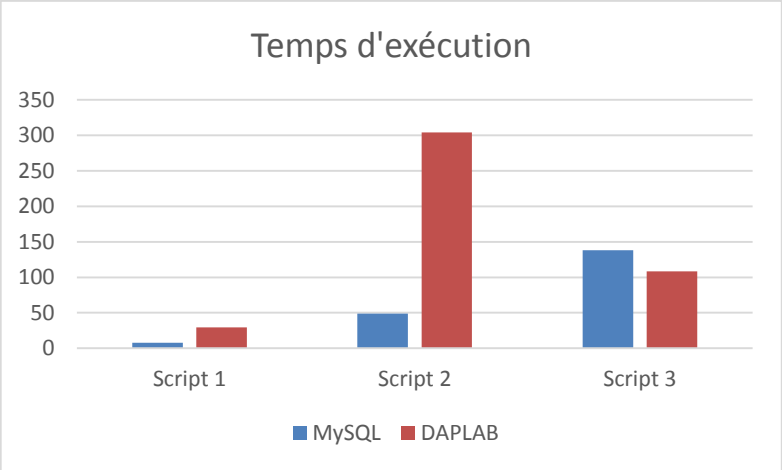


Figure 32 : Temps d'exécution des scripts

On notera que les résultats obtenus sur l'architecture MySQL sont très instables. En effet, le même script lancé à quelques secondes de différences peut avoir des résultats totalement différents. De plus, les requêtes sur le troisième script (très grosse quantité de données) n'aboutissent parfois pas du tout.

Nous n'avons donc qu'une seule valeur utilisable de référence pour l'exécution du dernier script.

5.4. Conclusion

Les tests effectués dans ce projet ont démontré que l'intégration d'Hadoop dans le processus de récolte de données et de génération de statistiques était possible. Le fonctionnement de chacune des étapes, l'importation des données, la création de scripts générant les données de statistiques et la génération de graphique ont été validés.

Les tests de performance démontrent que l'architecture actuelle était largement plus performante sur des requêtes avec de petites quantités de données, mais n'était plus adaptée à des requêtes utilisant de grandes quantités de données.

6. Bonnes pratiques

6.1. Introduction

Ce chapitre a pour but de recenser les bonnes pratiques à suivre qui ont été apprises tout au long du projet, afin d'aider la migration de futurs projets similaires, et également quelques bonnes pratiques à suivre pour le développement de scripts sur de grandes quantités de données.

6.2. Réaliser un projet de migration

Les prochaines sections décrivent les étapes à réaliser lors de la migration d'un système existant vers une infrastructure utilisant un cluster Hadoop.

6.2.1. Etude des données

La première étape consiste à étudier le type, la position et le volume de données que l'on souhaite intégrer dans un cluster Hadoop. Un autre point important est de connaitre l'évolution de la quantité de données sur le système. Si le volume de données est peu important, ou si la taille des données est réparties en fichiers de petite taille ne pouvant pas être concaténés, ou n'ayant pas de lien entre eux pour leur utilisation, l'architecture Hadoop n'est pas adaptée.

Dans le cas du NetObservatory, les quantités de données étaient déjà relativement importantes, avec plusieurs centaines de giga-octets. La quantité de données pourra évoluer de manière exponentielle si les récoltes de données sont étendues à de nouvelles régions. L'intégration des données dans Hadoop permet ainsi de préparer le terrain à l'évolution du projet.

6.2.2. Etude de l'interaction des données

Il est important de schématiser un processus complet avec les différentes interactions nécessaires entre les données elles-mêmes, et les différents programmes ou applications y ayant accès. Ce processus permettra de déterminer les étapes et les modifications nécessaires pour la migration.

Il est également intéressant de connaître l'utilisation finale des données.

6.3. Effectuer la migration par étapes

Il est conseillé d'effectuer la migration des données par étapes, de valider le bon fonctionnement de l'étape réalisée pour obtenir de bonnes performances, puis de passer à la prochaine étape.

Le projet NODaBOp étant principalement un projet de découverte et d'apprentissage, il s'est concentré sur la migration des données et le traitement des données pour générer des graphiques.

6.4. Développement

Comme pour la réalisation d'un projet de migration, le développement pour une architecture distribuée devrait être effectué par étapes.

Dans le cas particulier du projet NetObservatory, tous les scripts ont été développés en Scala, qui met à disposition la console `spark-shell`. L'avantage de cette solution est qu'il est possible de tester le fonctionnement d'un script complet étapes par étapes plutôt que dans son intégralité.

Dès lors, pour le développement d'un script Scala, il est conseillé de travailler tout d'abord sur un petit set de donnée représentatif de la totalité des données, et sur une infrastructure réduite (Comme la machine virtuelle HortonWorks ici), afin de valider le fonctionnement du script. A l'aide de la console spark-shell, il est possible de connaître et d'afficher l'état des données à tout moment.

Une fois chacune des étapes d'un script validées depuis la console, il est désormais possible d'exécuter la totalité du script sur une petite quantité de données sur le cluster, pour valider que le script fonctionne également sur celui-ci, puis on passe sur de grandes quantités de données.

6.5. Conclusion

Les bonnes pratiques énoncées dans ce chapitre permettent de connaitre les étapes importantes afin de réaliser un projet de migration similaire. Les points-clé de ce genre de migration sont l'analyse des données, recenser les programmes et applications les utilisant et d'établir un processus de migration par étapes.

7. Conclusion

7.1. Conclusion du projet

Ce projet a permis de réaliser et documenter une analyse minutieuse de l'écosystème Hadoop, qui permet de connaitre son fonctionnement, ainsi que les différents outils et leur utilisation. Un processus de migration a permis de décrire et définir comment intégrer les différents éléments du projet NetObservatory avec Hadoop et plusieurs graphiques ont pu être générés sur cette nouvelle architecture afin de valider le bon fonctionnement de la chaine, et prouver la faisabilité du projet. Finalement, des tests de performances ont été réalisés sur les deux architectures afin de déterminer laquelle est la plus optimale au niveau des performances.

Les tests de performances ont démontré que l'infrastructure actuelle offre de meilleures performances lors du traitement sur les petites quantités de données. Cependant, la nouvelle architecture offre une *scalability* et est plus performante sur de très grandes quantités de données. De plus, le nouveau processus offre de meilleurs résultats sur le temps de génération du graphique, car les données ont été travaillées et préparées directement lors de la récolte.

Ce projet énumère également plusieurs bonnes pratiques à suivre pour la réalisation d'un projet de migration de données vers une infrastructure Hadoop.

7.2. Difficultés rencontrées

Quelques difficultés ont été rencontrées durant ce projet et ce sont elles qui l'ont rendu plus intéressant et m'ont permis d'acquérir de nouvelles connaissances.

7.2.1. Accès à la base de données MySQL

Le serveur de bases de données de test utilisé pour ce projet (BD05) ne comportait pas la totalité des tables et n'était pas accessible depuis le réseau HEIA-FR. Elle a été rendue accessible d'abord avec un bypass du Firewall uniquement pour l'adresse IP de la machine du laboratoire et la nécessité d'entrer une route statique, puis par la création d'une entrée NAT statique avec l'adresse `160.98.30.240`.

Elle a été temporairement rendue inaccessible et l'installation d'un serveur MySQL temporaire a été mise en place sur l'infrastructure de test à partir des fichiers mysqldump de backup.

7.2.2. Corruption des fichiers mysqldump

Certains fichiers dump de backup étaient corrompus et n'ont donc pas pu être intégrés dans les bases de données. Cela a permis la création de scripts de génération de graphiques adaptatifs au manque possible de certaines données.

7.2.3. Architecture réseau de la HEIA-FR

L'architecture réseau qui a été utilisée lors de ce projet comportait de nombreux Firewalls, et de ce fait, la communication entre les différents équipements a été difficile à mettre en place. En effet, l'ordinateur du laboratoire avait accès à la base de données MySQL et au DAPLAB, mais les deux éléments ne pouvaient pas directement communiquer.

Avec l'aide de Benoit Perroud, des configurations spécifiques, décrites précisément dans le chapitre 4.2 de l'Implémentation, ont permis d'établir une communication entre les différents éléments.

7.2.4. Accès au DAPLAB

Le cluster Hadoop DAPLAB de la HEIA-FR étant actuellement en cours de mise en place, il a subit plusieurs maintenance au cours du projet, nécessitant des coupures de service. La plus grande partie du développement a été effectuée sur une machine virtuelle Hortonworks installée en local.

7.2.5. Evolution des technologies

Hadoop étant une technologie jeune, elle reste encore en constante évolution. Il a parfois été difficile de trier les données concernant les anciennes versions.

7.3. Perspectives futures

Le projet a permis de valider la faisabilité de la première partie d'une migration de MySQL vers Hadoop. La Figure 33 permet de mieux comprendre la chaine de traitement réalisée lors de ce projet.

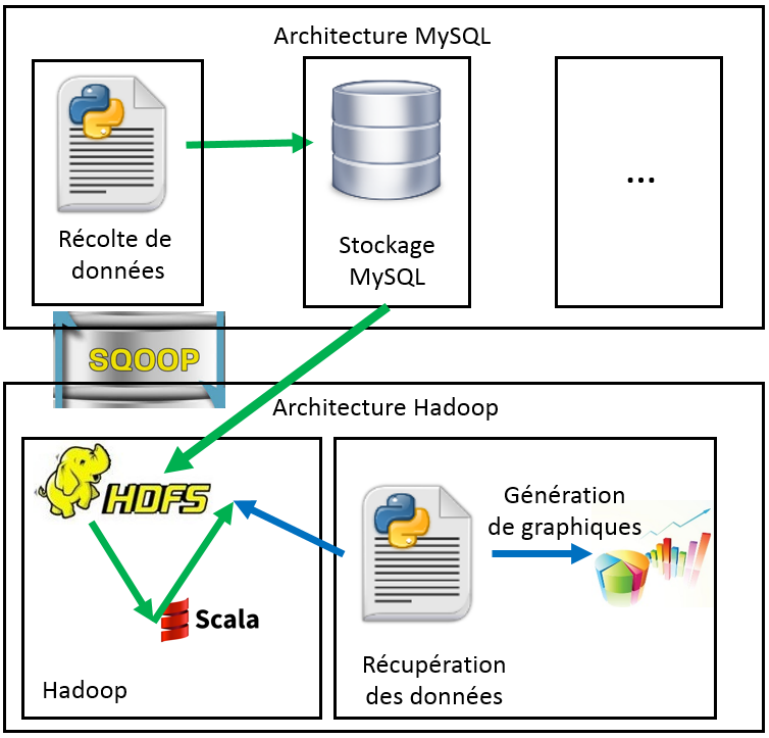


Figure 33 : Intégration par étapes

7.3.1. Importation des données

La méthodologie décrite dans le chapitre 4.3.3 permettra de simplifier l'importation de la totalité des tables MySQL vers Hadoop.

7.3.2. Scripts de statistiques

Trois scripts de génération de statistiques ont pu être créés, afin de valider la faisabilité. Il existe actuellement une vingtaine de scripts qui devront également être traduits pour Hadoop, et éventuellement retravaillés. L'annexe 4 de ce document donne quelques pistes d'améliorations pour certains de ces scripts.

7.3.3. Migration de la récolte des données

Le chapitre 3.3 de la conception a décrit deux méthodes pour intégrer les données récoltées par les scripts de récoltes dans Hadoop. Une suite possible à donner à ce projet est de modifier les scripts de récoltes de données de manière à créer directement des fichiers utilisables par Hadoop et s'affranchir ainsi de la base de données MySQL.

7.3.4. Extension du projet

Le projet NetObservatory tend vers l'extension des récoltes de données à d'autres régions, notamment l'Europe. L'architecture Hadoop mise en place sur le DAPLAB permettra de faciliter cette transition et devrait alors profiter de tous les avantages d'une architecture distribuée pour traiter de grandes quantités de données.

7.3.5. Création de nouveaux outils

De nouveaux outils de statistiques pourront être imaginés, car le temps de traitement des données ne sera désormais plus une inquiétude. De plus, les données statistiques utilisées pour générer les graphiques forment de nouvelles sources d'information. Ces dernières permettront de créer de nouveaux graphiques, par exemple sous forme d'histogrammes.

7.4. Remerciements

Je tiens tout particulièrement à remercier les personnes qui, de près ou de loin, m'ont offert de leur temps afin de m'encadrer, renseigner et partager quelques points de vue durant ce projet :

M. François Buntschu et Mme Houda Chabbi, Professeurs en Télécommunications à la HEIA-FR, qui m'ont suivi et supervisé durant tout le travail et auprès desquels j'ai pu bénéficier d'une grande expérience dans les domaines des bases de données et du réseau.

M. Benoit Perroud, Senior Software Engineer chez Verisign et responsable du Data Analysis and Processing Lab à la HEIA-FR, pour ses conseils pour l'intégration du projet NetObservatory dans Hadoop et son aide pour l'accès au DAPLAB.

MM Pierre-Alain Mettraux et Robert Van Kommer, Professeur à Ecole des métiers de Fribourg, respectivement conseiller scientifique chez Alliance, qui m'ont suivi et supervisé durant ce travail en tant qu'experts.

MM. Romain Froidevaux et Loïc Gremaud, étudiants à la HEIA-FR, pour leurs conseils et leurs idées d'améliorations du projet.

Remerciements particuliers à Mme Yvette Rossier pour la relecture du présent document.

7.5. Conclusion personnelle

La totalité du projet a été très intéressante pour moi, car elle m'a permis d'apprendre de nombreuses nouvelles technologies, en particulier l'apprentissage du développement de scripts fonctionnant sur une architecture distribuée.

Les différents problèmes rencontrés m'auront permis d'apprendre plus précisément l'utilisation des outils Linux tels que SSH et SCP, mais également d'apprendre à trouver des solutions et à réfléchir et déterminer plus précisément à la cause des problèmes, et à trouver des solutions intermédiaires, notamment dans le cas des problèmes de communications entre les différents éléments du réseau.

L'apprentissage du nouveau langage de programmation en vogue, Scala a été un grand plus. Il s'agit d'un langage facile à prendre en main et très intuitif, à l'image du Python.

Je suis heureux d'avoir pu faire un pas dans ce nouveau monde qu'est celui des BigData, en particulier avec l'apprentissage de l'écosystème Hadoop.

7.6. Déclaration sur l'honneur

Je, soussigné, David Rossier, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

Fribourg, le 10.07.2015
David Rossier

8. Contenu du CD

Chemin	Contenu
/README.pdf	Liste le contenu du CD
/Administration/	Donnée du projet
/Documentation/	Documentation du projet
/Documentation/Minutes	PV et invitations des séances
/Documentation/Planification.pdf	La planification du projet
/Documentation/JDB.pdf	Journal de bord
/Documentation/Final.pdf	Documentation finale au format PDF
/Documentation/Final.docx	Documentation finale au format DOCX
/Documentation/CahierDesCharges.pdf	Cahier des charges du projet
/Documentation/Scripts/	Informations sur les différents scripts NetObservatory
/Documentation/SQL/	Informations sur les tables et requêtes SQL utilisées pour les trouver
/Documentation/MapReduce.pdf	Complément d'information sur les Map/Reduce
/Documentation/Tests/	Contient les résultats des tests fonctionnels et des tests de performance
/NetObservatory/	Tous les fichiers liés au NetObservatory
/NetObservatory/Old/	Anciens scripts et informations collectées
/NetObservatory/New/	Scripts développés lors du projet

9. Sources

9.1. Analyse

[AN01] Wiki "Apache Hadoop",
http://wiki.apache.org/hadoop/ [juin 2015]

[AN02] Dr. Dobbs : Hadoop "The lay of the Land", Tom White
http://www.drdobbs.com/database/hadoop-the-lay-of-the-land/240150854 [juin 2015]

[AN03] Dr. Dobbs : Hadoop "Writing and Running Your First Project", Tom White
http://www.drdobbs.com/database/hadoop-writing-and-running-your-first-pr/240153197 [juin 2015]

[AN04] Dr. Dobbs : Hadoop "Writing Hadoop Programs in Python ", Gaston Hillar
http://www.drdobbs.com/database/pydoop-writing-hadoop-programs-in-python/240156473 [juin 2015]

[AN05] Cloudera, Hadoop "Guide to python frameworks for hadoop ", Uri Laserson
http://blog.cloudera.com/blog/2013/01/a-guide-to-python-frameworks-for-hadoop/ [juin 2015]

[AN06] Apache, Hadoop "Documentation Hadoop v2.7", Uri Laserson
http://hadoop.apache.org/docs/current/ [juin 2015]

[AN07] Importing MySQL data into Hadoop Cluster using Sqoop, Severalnines
http://severalnines.com/blog/archival-and-analytics-importing-mysql-data-hadoop-cluster-using-sqoop [juin 2015]

[AN08] Sqoop documentation
http://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html#_introduction [juin 2015]

[AN09] HortonWorks Website
http://hortonworks.com/ [juin 2015]

[AN10] HDFS Documentation on Apache v1.0
http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html

[AN11] Apache Software Foundation, "Hadoop v2.7 Documentation"
https://hadoop.apache.org/docs/r2.7.0/ [juin 2015]

[AN12] Apache Software Foundation, "Hive Documentation officielle"
https://cwiki.apache.org/confluence/display/Hive/Home

[AN13] Différents auteurs, "Présentations de Hive"
https://cwiki.apache.org/confluence/display/Hive/Presentations [juin 2015]

[AN14] Présentation, "Introduction to Hive", Jeff Hammerbatch
http://fr.slideshare.net/jhammerb/20081030linkedin [juin 2015]

[AN15] Différences entre Hadoop 1 et 2, "Dezyre"
http://www.dezyre.com/article/hadoop-2-0-yarn-framework-the-gateway-to-easier-programming-for-hadoop-users/84 [juin 2015]

[AN16] Apache Software Foundation, "Apache Spark"

http://spark.apache.org/ [juin 2015]

[AN17] Apache Software Foundation, "Apache HBase"

http://wiki.apache.org/hadoop/Hbase [juin 2015]

[AN18] Tom White, "Hadoop The Definitive Guide" [juillet 2015]

9.2. Implémentation

[IM1] EPFL, "Scala Language API "

http://www.scala-lang.org/api/current/#package [juillet 2015]

[IM2] Convertir clé Linux

https://techtuts.info/2014/06/convert-ppk-id_rsa-linux/ [juillet 2015]

[IM3] Tunnels autossh

https://raymii.org/s/tutorials/Autossh_persistent_tunnels.html [juillet 2015]

[IM4] Connexion SSH en Python

http://stackoverflow.com/questions/6188970/how-to-make-a-ssh-connection-with-python [juillet 2015]

[IM5] Connexion SSH avec clé privée

http://stackoverflow.com/questions/8382847/how-to-ssh-connect-through-python-paramiko-with-public-key [juillet 2015]

[IM6] Execution d'une commande externe en python

http://stackoverflow.com/questions/89228/calling-an-external-command-in-python [juillet 2015]

[IM7] Configuration SSH

http://nerderati.com/2011/03/17/simplify-your-life-with-an-ssh-config-file/ [juillet 2015]

[IM8] Apache Software Foundation, " Spark Guide "

https://spark.apache.org/docs/latest/programming-guide.html#example [juillet 2015]

[IM9] Daniel Westheide, " Spark Guide "

http://danielwestheide.com/blog/2012/12/19/the-neophytes-guide-to-scala-part-5-the-option-type.html [juillet 2015]

[IM10] Date & Time in Scala

http://stackoverflow.com/questions/24694849/how-to-get-the-current-date-without-time-in-scala [juillet 2015]

[IM11] Scala coalesce

http://stackoverflow.com/questions/24371259/how-to-make-saveastextfile-not-split-output-into-multiple-file [juillet 2015]

[IM12] Cloud Academy, " Sqoop helper "

http://cloudacademy.com/blog/big-data-getting-started-with-hadoop-sqoop-hive/ [juillet 2015]

[IM13] R Installation

http://cran.r-project.org/bin/linux/ubuntu/README [juillet 2015]

http://stackoverflow.com/questions/4723607/rpy2-installation-on-ubuntu [juillet 2015]

http://ggplot2.org/ [juillet 2015]

http://www.r-bloggers.com/installing-r-packages/ [juillet 2015]

[IM14] MySQL Access

https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql [juillet 2015]

[IM15] Putty

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html [juillet 2015]

[IM16] Apache Foundation, "Oozie Overview"

http://oozie.apache.org/docs/4.2.0/CoordinatorFunctionalSpec.html#a1._Coordinator_Overview [juillet 2015]

[IM17] Scala Helper for reduce

http://stackoverflow.com/questions/25158780/difference-between-reduce-and-foldleft-fold-in-functional-programming-particula [juillet 2015]

10. Table des figures

10.1. Sources

[1] http://files.dezyre.com/images/blog/Hadoop_2.0_and_YARN_Advantages_over_Hadoop_1.0_2.png

[2] https://hadoop.apache.org/docs/r2.7.0/hadoop-project-dist/hadoop-hdfs/images/hdfsarchitecture.png

[3] https://hadoop.apache.org/docs/r2.7.0/hadoop-project-dist/hadoop-hdfs/images/hdfsdatanodes.png

[4] http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/yarn_architecture.gif

[5] http://www.it-stammtisch-darmstadt.de/vortraege/unterlagen/vortrag-big-data.pdf

[6] https://cwiki.apache.org/confluence/display/Hive/HCatalog+UsingHCat

[7] https://www.netobservatory.ch/static/static_files/nora/archives/2013-Q2_NORA.pdf

11. Glossaire

Abbréviation	Définition
API	Application Programming Interface : Ensemble de classes, méthodes ou fonction qui offre des services à d'autres logiciels.
BLOB	Binary Large OBject : Les grands objets binaires stockées notamment dans les bases de données.
CMS	Content Management System : Système de gestion de contenus, principalement connu dans le domaine du Web pour faciliter la gestion des contenus Web aux utilisateurs
CSV	Comma Separated Value : Format de fichier séparé par des virgules
DNS	Domain Name System : Service permettant de traduire les adresses IP en nom de domaine et inversement.
HDFS	Hadoop Distributed File System : Le système de fichier utilisé par Hadoop
IP	Internet Protocol
JDBC	Java Database Connector : Connecteur Java qui permet de se connecter à une base de données et y effectuer des requêtes SQL
NAT	Network Address Translation : Service qui consiste à traduire une adresse IP par une autre en traversant un équipement réseau.
NORA	NetObservatory Report and Analysis : Le rapport généré régulièrement par le projet NetObservatory pour faire un point sur la sécurité de l'internet suisse.
RAM	Random Access Memory : Mémoire vive des machines informatiques
RDBMS	Relational DataBase Management System : Système de gestion de base de données relationnelles, comme MySQL
RDD	Resilient Distributed Dataset. Dans le contexte de Spark, il s'agit d'une collection d'éléments qui sont distribués sur plusieurs machines d'un cluster Hadoop
RPC	Remote Procedure Call : Protocole réseau permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications
SCP	Secure Copy Protocol : protocole informatique permettant d'effectuer un transfert de fichiers sécurisés en utilisant le protocole de communication SSH
SGBD	Système de Gestion de bases de données
SQL	Structured Query Language : Langage de programmation normalisé servant à effectuer des requêtes sur les bases de données
SSH	Secure SHell est un programme informatique et un protocole de communication sécurisé.
TCP	Transmission Control Protocol : Protocole de transport fiable.
YARN	Yet Another Resource Negotiator : Technologie de gestion de cluster utilisée par Hadoop 2.

12. Annexes

1) Cahier des charges et Planification
2) Tableau des scripts de récolte
3) Tableau des scripts de statistiques
4) Informations sur les scripts de statistiques
5) Détails des champs des tables MySQL
6) Taille des tables MySQL et requêtes SQL
7) Map/Reduce : Exemple et complément d'informations
8) Tests fonctionnels
9) Tests de performance

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

TIC – Filière Télécommunications

Orientation Réseaux et Sécurité

Projet de bachelor

01 juin 2015

Utilisation du NoSQL pour l'analyse des données du NetObservatory

Annexe 1 : Cahier des charges

Version 1.0

Étudiant :	David Rossier
Mandaté par :	NetObservatory
Supervisé par :	François Buntschu Houda Chabbi Drissi
Experts :	Pierre-Alain Mettraux Robert Van Kommer

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Historique du document

Version	Auteur	Description des modifications	Date
0.1	David Rossier	Création du document DRAFT 1	26.05.2015
0.2	David Rossier	Contexte, objectifs, liste des tâches	27.05.2015
0.3	David Rossier	Version DRAFT envoyée aux superviseurs	28.05.2015
0.4	David Rossier	Correction Mme Chabbi	29.05.2015
1.0	David Rossier	Correction M.Buntschu	01.06.2015

Table des matières

Cahier des charges

1. Contexte

La sécurité des infrastructures informatiques dépend de plusieurs facteurs, soit de la configuration du réseau et des éléments de protection tels que les firewalls, mais surtout de la fiabilité des applications utilisées par les serveurs et les clients (ainsi que des systèmes d'exploitation). Tous les logiciels du marché présentent des vulnérabilités qui peuvent être exploitées de façons malveillantes par des hackers, soit pour voler de l'information, soit pour déployer des réseaux de bots.

Ces infrastructures sont le plus souvent connectées à Internet et nécessitent un nom de domaine pour pouvoir fonctionner. Dans le cadre du projet de recherche NetObservatory mené à la HEIA-FR, différentes données sont collectées sur le réseau Internet Suisse, telles que le type et les versions logicielles utilisées par les serveurs web. Diverses statistiques sont ensuite générées comme, par exemple, le top 10 des outils web (CMS) les plus utilisés ou encore le top 10 des domaines les plus vulnérables.

Le projet NetObservatory mené à la HEIA-FR depuis 2009, a permis de développer des outils pythons de récoltes des données sur le réseau Internet Suisse dans le cadre de plusieurs projets de bachelor. Comme on peut le voir dans la figure 1, ces scripts enregistrent les données récoltées dans des tables MySQL, en indiquant la date de la récolte. Les données collectées représentent aujourd'hui plus de 400GB d'informations.

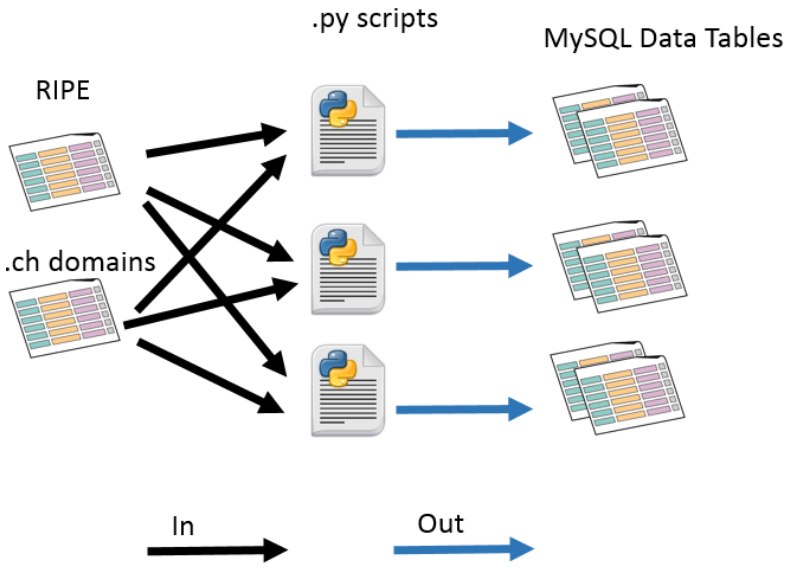


Figure 1 : Récolte de données

Ces données sont ensuite utilisées pour générer des graphiques de statistiques, qui sont utilisées principalement à l'heure actuelle pour la génération du rapport NORA (NetObservatory Report & Analysis). Le rapport NORA permet d'évaluer le niveau de sécurité offert par les infrastructures suisses et était publié plusieurs fois par année jusqu'en 2013. Ces statistiques sont générées à l'aide de scripts python qui interrogent la base de données MySQL et génèrent des graphiques (Figure 2).

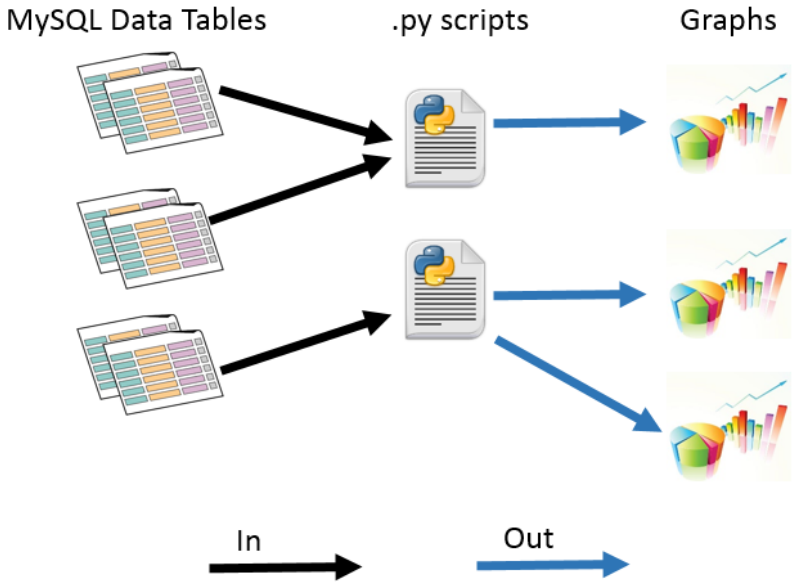


Figure 2 : Génération des graphiques

Malheureusement, la génération des statistiques au travers des requêtes sur la base de données devient très coûteuse en temps et en ressources. De plus, les mandants du projet aimeraient passer à des générations de données plus fréquentes, et également à la demande. Cela représentera donc d'avantage de données et l'architecture MySQL actuelle semble atteindre ses limites.

Le but de ce projet est de proposer une migration de cette base de données MySQL vers une base de données NoSQL (tel que HBase de Hadoop) afin d'optimiser l'utilisation de ces données.

2. Objectifs

À la fin du projet, le mandant aura à disposition :

1. Une analyse de l'écosystème Hadoop et des outils qu'il offre pour choisir les plus adéquats pour l'utilisation attendue dans le NetObservatory.
2. Une modélisation des données du NetObservatory dans Hadoop en vue des statistiques à mener ainsi qu'une approche pour la migration des données de MySQL vers Hadoop.
3. Une implémentation de quelques outils statistiques avec le paradigme MAP/Reduce sur cette nouvelle architecture et une comparaison des performances avec l'ancienne architecture.

Par ailleurs, un objectif secondaire pourra être étudié et / ou réalisé lors du projet :

- Etude des possibilités de Data Mining sur les données du NetObservatory

3. Liste des tâches

Les tâches ci-dessous seront réalisées tout au long du projet et sont réparties en quatre phases principales : Analyse de l'état de l'art, conception, implémentation et tests.

3.1. Analyse de l'état de l'art

Afin de mener à bien ce projet, plusieurs études seront réalisées :

- État actuel de NetObservatory
  - Contenu et format des tables MySQL
- Etude des scripts pythons
- Etude du fonctionnement python – Hadoop
- Etude de Hadoop
  - HDFS
  - Différents outils et possibilités
  - Bases de données NoSQL
  - Transfert de données de MySQL vers Hadoop

3.2. Conception

Il sera ensuite nécessaire de concevoir une nouvelle architecture du fonctionnement du système. Les tâches suivantes seront effectuées :

- Modélisation des données dans Hadoop pour l'utilisation adéquate.
- Intégration d'Hadoop dans l'architecture NetObservatory selon les résultats de l'analyse
  - Stockage MySQL & Traitement Hadoop
  - Stockage & Traitement Hadoop

3.3. Implémentation

L'implémentation fera l'objet de plusieurs parties, et dépendront des résultats de l'analyse et de la conception

- Installation et configuration de l'environnement Hadoop sur le cluster de l'école
- Transfert des données MySQL dans Hadoop
- Mise à jour des scripts de collecte de données pour modifier la destination dans Hadoop (stockage et traitement dans Hadoop)
- Choix des outils statistiques à implémenter en utilisant les outils proposés par Hadoop

3.4. Tests et validation

Une fois l'implémentation terminée, et tout au long du projet, il sera nécessaire de valider le bon fonctionnement des outils implémentés. Les tests suivants seront effectués :

- Tests fonctionnels
  - Outils de migration MySQL vers Hadoop
  - Scripts de génération de graphiques (NORA)
  - Scripts d'intégration
- Tests de performances et analyse des résultats
  - Temps de génération des graphes
  - Comparaison nouvelle architecture Hadoop – ancienne architecture MySQL

4. Organisation

4.1. Gestion des documents

Tous les rendus de documents (rapports, invitations aux séances, procès-verbaux, etc…) ainsi que les codes seront déposés sur le git de l'école, consultable à l'adresse suivante :

https://forge.tic.eia-fr.ch/git/david.rossier/nodabop

4.2. Planification

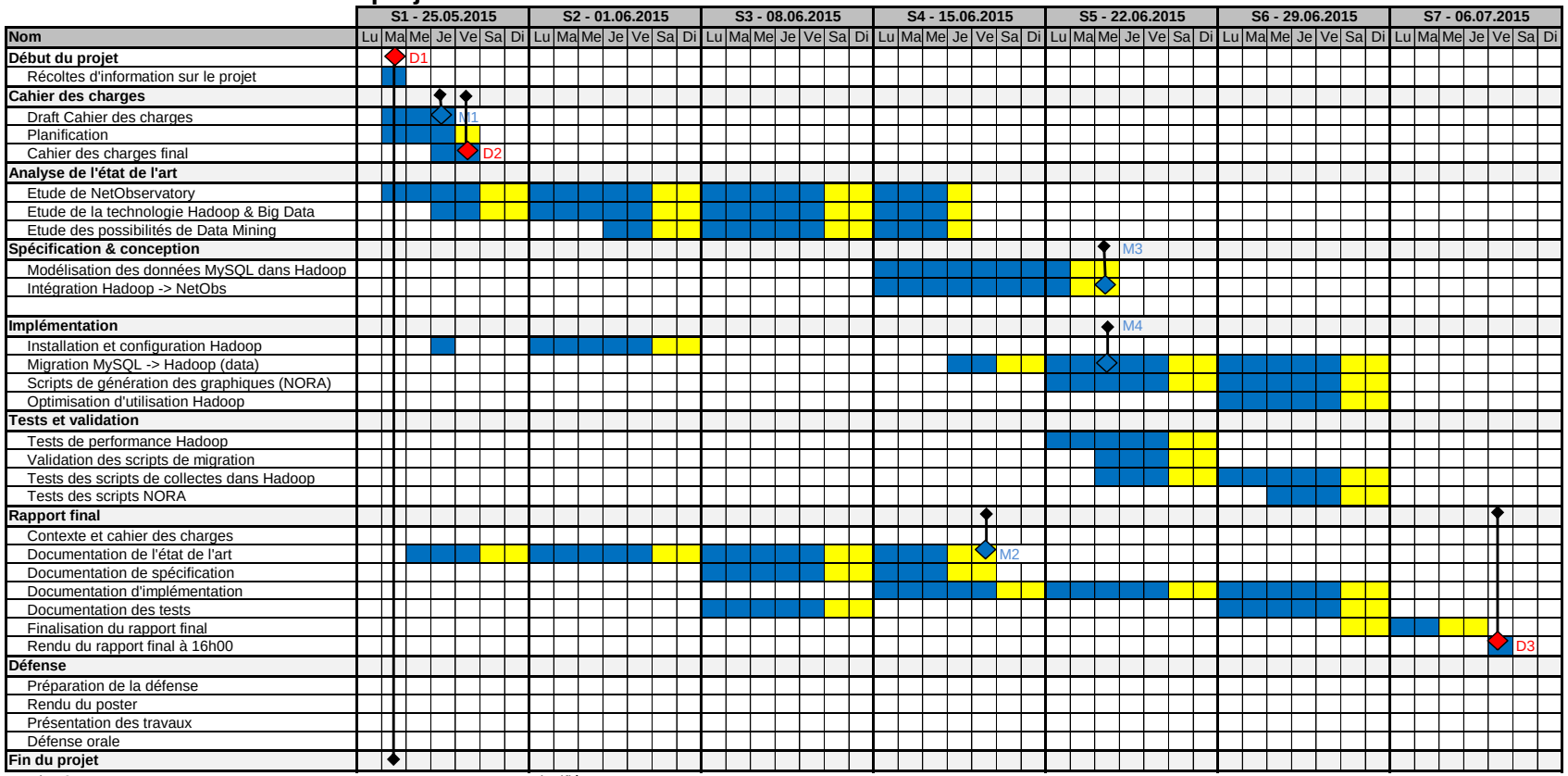
Afin de mener à bien le projet et d'avoir une vision globale du temps à disposition, une planification détaillée a été réalisée. Une marge de sécurité a été prévue pour certains jalons en cas de soucis.

La planification des tâches à effectuer est disponible dans l'annexe 1.

Fribourg, le 10/07/2015

David Rossier

Annexe 1. Planification du projet



Version 2
Date : 17.06.2015

Légendes: Temps planifié: Planifié, Marge, DeadLine, MileStone; Réel

D1 Début du projet
D2 Rendu du cahier des charges
D3 Rendu du rapport final
D4 Défense orale
D5 Fin du projet

M1 Draft cahier des charges
M2 Rendu de l'analyse
M3 Rendu des spécifications
M4 Démonstration application simple

Tools (1)

Tools	IN	OUT	Description
scan	• dns • blacklist	• scan_os • scan_services	Scan toutes les IPv4 présentes dans la table DNS
ssl_discover	• dns • blacklist	• ssl_discover	Contrôle tous les certificats SSL accessible
nvd	-	• nvd_cvss	Charge toutes les vulnérabilités connues
bgp_scan	• lookkingGlass • ripe	• autoSystem • annonceV4 • annonceV6 • routeIPv4 • routeIPv6 • Voisins • bgpresults	Récolte les tables de routages de tous les réseaux Suisse depuis plusieurs « looking glass »
smtp_discover	• dns	• smtp_discover	Validation des configurations des serveurs DNS des .ch
dns	• whois • blacklist	• dns	Récolte les infos A, AAAA, MX, NS, etc d'un échantillons de domaines .ch
reverse_dns	• ripe	• reverse_dns	Effectue une résolution DNS inverse sur toutes les IP Suisses.

netobservatory

Tools (2)

Tools	IN	OUT	Description
reverse_dns	• ripe	• reverse_dns	Effectue une résolution DNS inverse sur toutes les IP Suisses.
dns_discover	• dns	• dns_discover	Teste les configurations DNS (récursivité, glue record, transfert de zone, etc.)
dnssec_discover	• whois	• dnssec_validation • root • 1ld • 2ld • 3ld	Validation des configurations DNSSec
RIPE	-	• ripe	Charge toutes les IP (v4 et v6) attribuées à la Suisse
whois	-	• whois	Récolte tous les .ch sur la base whois de Switch
Web_fingerprinting	• dns • blacklist	• web_fingerprint • web_fingerprint_full	Récolte la 1ère page de tous les www.xxxx.ch de Suisse

ID	Tools	Table IN	Projection	Filtre	Remarques
1	dnssec_digest	2ld	"ds%d_digest_type"	GROUP BY "ds%d_digest_type"	%d = chiffre 1 à 5
202	statistics	ripe	count(distinct as_number)		
		whois	count(distinct holder)		
		dns	count(distinct ip)	WHERE type = ns, mx, a	
		web_fingerprint	count(distinct domain)		
2031	whois_stats	npa	distinct npa canton		
		whois	holder		
2023	whois_holders	whois	holder		
2041	hosting_stats	ripe	as_name ip_range		
		dns	ip	WHERE type = 'a' and ip_version='4'	
2042	ws_freq	scan_services	software count(1) nb	WHERE service like "http%" GROUP BY software ORDER BY nb DESC	
2051	build_dns_ns_freq	dns	fqdn domain	WHERE type='ns'	Utilisation de dns_ns_freq pour les calculs
2061	build_dns_mx_freq	dns	fqdn domain	WHERE type='mx'	Utilisation de dns_mx_freq pour les calculs
2071	os_year_vuln	nvd_vulns	substring(cve, 5, 4) year vendor count(distinct cve)	WHERE (vendor = 'microsoft' or vendor = 'linux' or vendor = 'apple') and part = 'o' and product not like '%%server%%' GROUP BY year, vendor ORDER BY vendor, year	
3011	cert_valid_freq	ssl_discover	count(distinct domain)	GROUP BY verif_error asc, valid desc	
3012	keylen_freq	ssl_discover_%date	keylength*8 count(distinct cert_fingerprint)	WHERE keylength > 0 GROUP BY keylength asc	
3013	sigalgo_freq	ssl_discover	sigalgo count(distinct cert_fingerprint)	WHERE sigalgo is not NULL group by sigalgo asc	
3014	sigalgo_evolution	ssl_discover	sigalgo count(distinct cert_fingerprint)	WHERE YEAR(not_before) = %dates% and (sigalgo = 'md5' or sigalgo = 'sha1') GROUP BY sigalgo asc	
3015	ssl_ciphers_freq	ssl_discover	%cyphers% count(distinct domain)	WHERE ciphers is not NULL GROUP BY %cyphers%	cypher définit dans le py
3016	issuer_freqs	ssl_discover	issuer count(distinct cert_fingerprint) nb	WHERE issuer is not NULL GROUP BY issuer ORDER BY nb DESC	
3021	webapp_stats	web_fingerprint_%date	%app% count(domain)	WHERE apps like '%app%'	%app% dans web_fingerprint/app.js
3022	cms_maps	npa	distinct npa canton		
		whois web_fingerprint	holder	JOIN whois.domain=web_fingerprint.domain WHERE and apps like '%app%'	apps = ['Joomla', 'TYPO3', 'WordPress']
3023	build_cms_vuln	web_fingerprint_%date	domain generator apps	WHERE app like typo3 or wordpress	Utilisation de cms_vuln pour le graphique
		nvd_vulns_%date	distinct version	WHERE product = 'wordpress' and version != ''	

		nvd_vulns_%date	distinct version	WHERE product = typo3 and version != ''	
3024	wordpress	web_fingerprint	count(1)	WHERE apps LIKE %ordpress% and fqdn LIKE www.%	
			count(1)	WHERE apps LIKE %ordpress% and fqdn LIKE www.% and generator LIKE '%ordpress %version%%'	version => Définies dans file
3031	build_ws_vuln	web_fingerprint	domain server		Utilisation de stats_webservers pour le graphique
		nvd_vulns	distinct version	WHERE (vendor = 'apache' and product like '%http_server%') WHERE product = 'nginx'	Deux requêtes du même champ sur la même table Filtre différent
4011	smtp_ssl	smtp_discover	count(distinct ip)	UNION SELECT count(distinct ip) FROM smtp_discover	
4012	smtp_stats	smtp_discover	server_type_p25 count(distinct ip) nb	WHERE server_type_p25 != 'null' GROUP BY server_type_p25 ORDER BY nb DESC	
4013	build_smtp_vuln	smtp_discover_%date	ip server_type_p25 server_version_p25	WHERE (server_type_p25 LIKE '%%exim%%' or server_type_p25 LIKE '%%sendmail%%')	Utilisation de smtp_vuln pour le graphique
5011	dns_recurs	dns_discover_%date	count(distinct ip)	WHERE recursive_tcp = 'yes' or recursive_udp = 'yes' UNION SELECT count(distinct ip) FROM dns_discover_%date	
5012	dns_ztrans	dns_discover_%date	zone_transfer count(distinct domain)	GROUP BY zone_transfer	
5013	dns_spf	dns	count(distinct domain)	WHERE type='txt' AND fqdn LIKE '%spf%' UNION SELECT count(distinct domain) from dns	
5014	dns_glue	dns_discover	count(distinct ip)	WHERE glue_record_validity = 'warning' UNION SELECT count(distinct ip) from dns_discover	
5021	dns_ipver	dns	count(distinct domain)		
		dns	type ip_version count(distinct domain)	WHERE type != 'txt' GROUP BY type, ip_version	
5031	dnssec_state	dnssec_validation_%date	secure existence_denied count(distinct domain)	GROUP BY secure, existence_denied	
5032	dnssec_time	dnssec_validation	total_time_dnssec total_time_dns		
5033	dnssec_algo	2ld	ds%d_algorithm count(1)	GROUP BY ds%d_algorithm	%d = chiffre 1 à 4
5034	dnssec_ksk_len	2ld	dnskey%d_key_size count(1)	WHERE dnskey%d_ksk_or_zsk = 'ksk' GROUP BY dnskey%d_key_size	%d = chiffre 1 à 6
5035	dnssec_zsk_len	2ld	dnskey%d_key_size count(1)	WHERE dnskey%d_ksk_or_zsk = 'zsk' GROUP BY dnskey%d_key_size	%d = chiffre 1 à 6
6011	scans_strange_ports	scan_services_%date	service count(distinct ip)	WHERE service in (%s) GROUP BY service	%s = listé dans script
6012	msds_stats	scan_services	software count(distinct ip) nb	WHERE port = '445' GROUP BY software ORDER BY nb DESC	

	Bien écrit pour MySQL	Remarques	Modification Dans Hadoop	Outil
dnssec_digest	NA			
statistics	NA			
whois_stats	Non	Requêtes séparées qui peuvent être JOIN	Dénormalisation Canton dans Whois	
whois_holders	Oui		Split le champ holder de whois par ; dénormalisation	
hosting_stats	Oui		Comparer IP avec Range IP dans Hadoop	
ws_freq	Oui		Recherche de version simplifiée dans requête	
build_dns_ns_freq	Non	Utilisation de table inutile	Split le champ fqdn pour récupérer l'host dns	
build_dns_mx_freq	Non	Utilisation de table inutile	Split le champ fqdn pour récupérer l'host mx	
os_year_vuln	Oui		Pas de modification	
cert_valid_freq	Oui		Pas de modification	
keylen_freq	Oui		Intégrer génération monthly dans Hadoop	
sigalgo_freq	Oui			
sigalgo_evolution	Non		Transformation pour moins de requêtes	
ssl_ciphers_freq	Oui		Transformation pour avoir les résultats finaux	
issuer_freqs	Oui		Travail de splits à faire sur les CNAME ; Dénormalisation	
webapp_stats	Non	Pas sûr (120 requêtes)	Meilleure organisation des données	
cms_maps	Oui		Split le champ holder de whois par ; dénormalisation	
build_cms_vuln	NA	Le script ne fonctionnait pas		
wordpress	Non	LIKE %ordpress% très couteux	Optimiser les différentes parties de la requête.	
build_ws_vuln	Oui		Traiter les données en plusieurs étapes	
smtp_ssl	Oui		Rien	
smtp_stats	Oui	Sorted	Meilleur traitement	
build_smtp_vuln	NA	Compliqué à comprendre		
dns_recurs	Oui	Chaque date : Get Recursive and total count	Hbase : infos triées par dates, récupérer directement	
dns_ztrans	Oui	Chaque date		
dns_spf	Oui	count spf and total count	Rien	
dns_glue	Oui	count glue and total count	Rien	
dns_ipver	Oui		Rien	
dnssec_state	Oui	Chaque date : DNSSEC State	Hbase : infos triées par dates, récupérer directement	
dnssec_time	Oui	Délai DNSSEC	Rien	
dnssec_algo	Oui	DNSSEC Algo	Parallélisation des requêtes	
dnssec_ksk_len	NA	Compliqué à comprendre		
dnssec_zsk_len	NA	Compliqué à comprendre		
scans_strange_ports	Oui	Count(distinct ip) group by services	Rien	
msds_stats	Oui	software, count(distinct ip)		

Annexe 5 : Détails des champs des tables MySQL					
DESCRIBE1ld_20140430, DESCRIBE 2ld_20140430, DESCRIBE 3ld_20140430					
Field	Type	Null	Key	Default	Extra
domain	varchar(256)	NO	PRI	NULL	
name	varchar(200)	YES		NULL	
ipv4	varchar(50)	YES		NULL	
ipv6	varchar(50)	YES		NULL	
rrsig1	longtext	YES		NULL	
rrsig1_name_assigned	varchar(100)	YES		NULL	
rrsig1_ttl	decimal(10,0)	YES		NULL	
rrsig1_klasse	varchar(20)	YES		NULL	
rrsig1_type	varchar(20)	YES		NULL	
rrsig1_covered_type	varchar(20)	YES		NULL	
rrsig1_algorithm	int(11)	YES		NULL	
rrsig1_label	int(11)	YES		NULL	
rrsig1_original_ttl	decimal(10,0)	YES		NULL	
rrsig1_expiration_time	bigint(20)	YES		NULL	
rrsig1_inception_time	bigint(20)	YES		NULL	
rrsig1_key_tag	int(11)	YES		NULL	
rrsig1_signers_name	varchar(100)	YES		NULL	
rrsig1_signatur	longtext	YES		NULL	
rrsig2	longtext	YES		NULL	
rrsig2_name_assigned	varchar(100)	YES		NULL	
rrsig2_ttl	decimal(10,0)	YES		NULL	
rrsig2_klasse	varchar(20)	YES		NULL	
rrsig2_type	varchar(20)	YES		NULL	
rrsig2_covered_type	varchar(20)	YES		NULL	
rrsig2_algorithm	int(11)	YES		NULL	
rrsig2_label	int(11)	YES		NULL	
rrsig2_original_ttl	decimal(10,0)	YES		NULL	
rrsig2_expiration_time	bigint(20)	YES		NULL	
rrsig2_inception_time	bigint(20)	YES		NULL	
rrsig2_key_tag	int(11)	YES		NULL	
rrsig2_signers_name	varchar(100)	YES		NULL	
rrsig2_signatur	longtext	YES		NULL	
rrsig3	longtext	YES		NULL	
rrsig3_name_assigned	varchar(100)	YES		NULL	
rrsig3_ttl	decimal(10,0)	YES		NULL	
rrsig3_klasse	varchar(20)	YES		NULL	
rrsig3_type	varchar(20)	YES		NULL	
rrsig3_covered_type	varchar(20)	YES		NULL	
rrsig3_algorithm	int(11)	YES		NULL	
rrsig3_label	int(11)	YES		NULL	
rrsig3_original_ttl	decimal(10,0)	YES		NULL	
rrsig3_expiration_time	bigint(20)	YES		NULL	
rrsig3_inception_time	bigint(20)	YES		NULL	
rrsig3_key_tag	int(11)	YES		NULL	
rrsig3_signers_name	varchar(100)	YES		NULL	
rrsig3_signatur	longtext	YES		NULL	
rrsig4	longtext	YES		NULL	
rrsig4_name_assigned	varchar(100)	YES		NULL	
rrsig4_ttl	decimal(10,0)	YES		NULL	
rrsig4_klasse	varchar(20)	YES		NULL	
rrsig4_type	varchar(20)	YES		NULL	
rrsig4_covered_type	varchar(20)	YES		NULL	
rrsig4_algorithm	int(11)	YES		NULL	
rrsig4_label	int(11)	YES		NULL	
rrsig4_original_ttl	decimal(10,0)	YES		NULL	
rrsig4_expiration_time	bigint(20)	YES		NULL	
rrsig4_inception_time	bigint(20)	YES		NULL	
rrsig4_key_tag	int(11)	YES		NULL	
rrsig4_signers_name	varchar(100)	YES		NULL	
rrsig4_signatur	longtext	YES		NULL	
rrsig5	longtext	YES		NULL	
rrsig5_name_assigned	varchar(100)	YES		NULL	
rrsig5_ttl	decimal(10,0)	YES		NULL	
rrsig5_klasse	varchar(20)	YES		NULL	
rrsig5_type	varchar(20)	YES		NULL	
rrsig5_covered_type	varchar(20)	YES		NULL	
rrsig5_algorithm	int(11)	YES		NULL	
rrsig5_label	int(11)	YES		NULL	

rrsig5_original_ttl	decimal(10,0)	YES		NULL	
rrsig5_expiration_time	bigint(20)	YES		NULL	
rrsig5_inception_time	bigint(20)	YES		NULL	
rrsig5_key_tag	int(11)	YES		NULL	
rrsig5_signers_name	varchar(100)	YES		NULL	
rrsig5_signatur	longtext	YES		NULL	
rrsig6	longtext	YES		NULL	
rrsig6_name_assigned	varchar(100)	YES		NULL	
rrsig6_ttl	decimal(10,0)	YES		NULL	
rrsig6_klasse	varchar(20)	YES		NULL	
rrsig6_type	varchar(20)	YES		NULL	
rrsig6_covered_type	varchar(20)	YES		NULL	
rrsig6_algorithm	int(11)	YES		NULL	
rrsig6_label	int(11)	YES		NULL	
rrsig6_original_ttl	decimal(10,0)	YES		NULL	
rrsig6_expiration_time	bigint(20)	YES		NULL	
rrsig6_inception_time	bigint(20)	YES		NULL	
rrsig6_key_tag	int(11)	YES		NULL	
rrsig6_signers_name	varchar(100)	YES		NULL	
rrsig6_signatur	longtext	YES		NULL	
rrsig7	longtext	YES		NULL	
rrsig7_name_assigned	varchar(100)	YES		NULL	
rrsig7_ttl	decimal(10,0)	YES		NULL	
rrsig7_klasse	varchar(20)	YES		NULL	
rrsig7_type	varchar(20)	YES		NULL	
rrsig7_covered_type	varchar(20)	YES		NULL	
rrsig7_algorithm	int(11)	YES		NULL	
rrsig7_label	int(11)	YES		NULL	
rrsig7_original_ttl	decimal(10,0)	YES		NULL	
rrsig7_expiration_time	bigint(20)	YES		NULL	
rrsig7_inception_time	bigint(20)	YES		NULL	
rrsig7_key_tag	int(11)	YES		NULL	
rrsig7_signers_name	varchar(100)	YES		NULL	
rrsig7_signatur	longtext	YES		NULL	
rrsig8	longtext	YES		NULL	
rrsig8_name_assigned	varchar(100)	YES		NULL	
rrsig8_ttl	decimal(10,0)	YES		NULL	
rrsig8_klasse	varchar(20)	YES		NULL	
rrsig8_type	varchar(20)	YES		NULL	
rrsig8_covered_type	varchar(20)	YES		NULL	
rrsig8_algorithm	int(11)	YES		NULL	
rrsig8_label	int(11)	YES		NULL	
rrsig8_original_ttl	decimal(10,0)	YES		NULL	
rrsig8_expiration_time	bigint(20)	YES		NULL	
rrsig8_inception_time	bigint(20)	YES		NULL	
rrsig8_key_tag	int(11)	YES		NULL	
rrsig8_signers_name	varchar(100)	YES		NULL	
rrsig8_signatur	longtext	YES		NULL	
rrsig9	longtext	YES		NULL	
rrsig9_name_assigned	varchar(100)	YES		NULL	
rrsig9_ttl	decimal(10,0)	YES		NULL	
rrsig9_klasse	varchar(20)	YES		NULL	
rrsig9_type	varchar(20)	YES		NULL	
rrsig9_covered_type	varchar(20)	YES		NULL	
rrsig9_algorithm	int(11)	YES		NULL	
rrsig9_label	int(11)	YES		NULL	
rrsig9_original_ttl	decimal(10,0)	YES		NULL	
rrsig9_expiration_time	bigint(20)	YES		NULL	
rrsig9_inception_time	bigint(20)	YES		NULL	
rrsig9_key_tag	int(11)	YES		NULL	
rrsig9_signers_name	varchar(100)	YES		NULL	
rrsig9_signatur	longtext	YES		NULL	
rrsig10	longtext	YES		NULL	
rrsig10_name_assigned	varchar(100)	YES		NULL	
rrsig10_ttl	decimal(10,0)	YES		NULL	
rrsig10_klasse	varchar(20)	YES		NULL	
rrsig10_type	varchar(20)	YES		NULL	
rrsig10_covered_type	varchar(20)	YES		NULL	
rrsig10_algorithm	int(11)	YES		NULL	
rrsig10_label	int(11)	YES		NULL	
rrsig10_original_ttl	decimal(10,0)	YES		NULL	

rrsig10_expiration_time	bigint(20)	YES		NULL	
rrsig10_inception_time	bigint(20)	YES		NULL	
rrsig10_key_tag	int(11)	YES		NULL	
rrsig10_signers_name	varchar(100)	YES		NULL	
rrsig10_signatur	longtext	YES		NULL	
rrsig11	longtext	YES		NULL	
rrsig11_name_assigned	varchar(100)	YES		NULL	
rrsig11_ttl	decimal(10,0)	YES		NULL	
rrsig11_klasse	varchar(20)	YES		NULL	
rrsig11_type	varchar(20)	YES		NULL	
rrsig11_covered_type	varchar(20)	YES		NULL	
rrsig11_algorithm	int(11)	YES		NULL	
rrsig11_label	int(11)	YES		NULL	
rrsig11_original_ttl	decimal(10,0)	YES		NULL	
rrsig11_expiration_time	bigint(20)	YES		NULL	
rrsig11_inception_time	bigint(20)	YES		NULL	
rrsig11_key_tag	int(11)	YES		NULL	
rrsig11_signers_name	varchar(100)	YES		NULL	
rrsig11_signatur	longtext	YES		NULL	
rrsig12	longtext	YES		NULL	
rrsig12_name_assigned	varchar(100)	YES		NULL	
rrsig12_ttl	decimal(10,0)	YES		NULL	
rrsig12_klasse	varchar(20)	YES		NULL	
rrsig12_type	varchar(20)	YES		NULL	
rrsig12_covered_type	varchar(20)	YES		NULL	
rrsig12_algorithm	int(11)	YES		NULL	
rrsig12_label	int(11)	YES		NULL	
rrsig12_original_ttl	decimal(10,0)	YES		NULL	
rrsig12_expiration_time	bigint(20)	YES		NULL	
rrsig12_inception_time	bigint(20)	YES		NULL	
rrsig12_key_tag	int(11)	YES		NULL	
rrsig12_signers_name	varchar(100)	YES		NULL	
rrsig12_signatur	longtext	YES		NULL	
rrsig13	longtext	YES		NULL	
rrsig13_name_assigned	varchar(100)	YES		NULL	
rrsig13_ttl	decimal(10,0)	YES		NULL	
rrsig13_klasse	varchar(20)	YES		NULL	
rrsig13_type	varchar(20)	YES		NULL	
rrsig13_covered_type	varchar(20)	YES		NULL	
rrsig13_algorithm	int(11)	YES		NULL	
rrsig13_label	int(11)	YES		NULL	
rrsig13_original_ttl	decimal(10,0)	YES		NULL	
rrsig13_expiration_time	bigint(20)	YES		NULL	
rrsig13_inception_time	bigint(20)	YES		NULL	
rrsig13_key_tag	int(11)	YES		NULL	
rrsig13_signers_name	varchar(100)	YES		NULL	
rrsig13_signatur	longtext	YES		NULL	
rrsig14	longtext	YES		NULL	
rrsig14_name_assigned	varchar(100)	YES		NULL	
rrsig14_ttl	decimal(10,0)	YES		NULL	
rrsig14_klasse	varchar(20)	YES		NULL	
rrsig14_type	varchar(20)	YES		NULL	
rrsig14_covered_type	varchar(20)	YES		NULL	
rrsig14_algorithm	int(11)	YES		NULL	
rrsig14_label	int(11)	YES		NULL	
rrsig14_original_ttl	decimal(10,0)	YES		NULL	
rrsig14_expiration_time	bigint(20)	YES		NULL	
rrsig14_inception_time	bigint(20)	YES		NULL	
rrsig14_key_tag	int(11)	YES		NULL	
rrsig14_signers_name	varchar(100)	YES		NULL	
rrsig14_signatur	longtext	YES		NULL	
rrsig15	longtext	YES		NULL	
rrsig15_name_assigned	varchar(100)	YES		NULL	
rrsig15_ttl	decimal(10,0)	YES		NULL	
rrsig15_klasse	varchar(20)	YES		NULL	
rrsig15_type	varchar(20)	YES		NULL	
rrsig15_covered_type	varchar(20)	YES		NULL	
rrsig15_algorithm	int(11)	YES		NULL	
rrsig15_label	int(11)	YES		NULL	
rrsig15_original_ttl	decimal(10,0)	YES		NULL	
rrsig15_expiration_time	bigint(20)	YES		NULL	

rrsig15_inception_time	bigint(20)	YES		NULL	
rrsig15_key_tag	int(11)	YES		NULL	
rrsig15_signers_name	varchar(100)	YES		NULL	
rrsig15_signatur	longtext	YES		NULL	
rrsig16	longtext	YES		NULL	
rrsig16_name_assigned	varchar(100)	YES		NULL	
rrsig16_ttl	decimal(10,0)	YES		NULL	
rrsig16_klasse	varchar(20)	YES		NULL	
rrsig16_type	varchar(20)	YES		NULL	
rrsig16_covered_type	varchar(20)	YES		NULL	
rrsig16_algorithm	int(11)	YES		NULL	
rrsig16_label	int(11)	YES		NULL	
rrsig16_original_ttl	decimal(10,0)	YES		NULL	
rrsig16_expiration_time	bigint(20)	YES		NULL	
rrsig16_inception_time	bigint(20)	YES		NULL	
rrsig16_key_tag	int(11)	YES		NULL	
rrsig16_signers_name	varchar(100)	YES		NULL	
rrsig16_signatur	longtext	YES		NULL	
rrsig17	longtext	YES		NULL	
rrsig17_name_assigned	varchar(100)	YES		NULL	
rrsig17_ttl	decimal(10,0)	YES		NULL	
rrsig17_klasse	varchar(20)	YES		NULL	
rrsig17_type	varchar(20)	YES		NULL	
rrsig17_covered_type	varchar(20)	YES		NULL	
rrsig17_algorithm	int(11)	YES		NULL	
rrsig17_label	int(11)	YES		NULL	
rrsig17_original_ttl	decimal(10,0)	YES		NULL	
rrsig17_expiration_time	bigint(20)	YES		NULL	
rrsig17_inception_time	bigint(20)	YES		NULL	
rrsig17_key_tag	int(11)	YES		NULL	
rrsig17_signers_name	varchar(100)	YES		NULL	
rrsig17_signatur	longtext	YES		NULL	
rrsig18	longtext	YES		NULL	
rrsig18_name_assigned	varchar(100)	YES		NULL	
rrsig18_ttl	decimal(10,0)	YES		NULL	
rrsig18_klasse	varchar(20)	YES		NULL	
rrsig18_type	varchar(20)	YES		NULL	
rrsig18_covered_type	varchar(20)	YES		NULL	
rrsig18_algorithm	int(11)	YES		NULL	
rrsig18_label	int(11)	YES		NULL	
rrsig18_original_ttl	decimal(10,0)	YES		NULL	
rrsig18_expiration_time	bigint(20)	YES		NULL	
rrsig18_inception_time	bigint(20)	YES		NULL	
rrsig18_key_tag	int(11)	YES		NULL	
rrsig18_signers_name	varchar(100)	YES		NULL	
rrsig18_signatur	longtext	YES		NULL	
rrsig19	longtext	YES		NULL	
rrsig19_name_assigned	varchar(100)	YES		NULL	
rrsig19_ttl	decimal(10,0)	YES		NULL	
rrsig19_klasse	varchar(20)	YES		NULL	
rrsig19_type	varchar(20)	YES		NULL	
rrsig19_covered_type	varchar(20)	YES		NULL	
rrsig19_algorithm	int(11)	YES		NULL	
rrsig19_label	int(11)	YES		NULL	
rrsig19_original_ttl	decimal(10,0)	YES		NULL	
rrsig19_expiration_time	bigint(20)	YES		NULL	
rrsig19_inception_time	bigint(20)	YES		NULL	
rrsig19_key_tag	int(11)	YES		NULL	
rrsig19_signers_name	varchar(100)	YES		NULL	
rrsig19_signatur	longtext	YES		NULL	
rrsig20	longtext	YES		NULL	
rrsig20_name_assigned	varchar(100)	YES		NULL	
rrsig20_ttl	decimal(10,0)	YES		NULL	
rrsig20_klasse	varchar(20)	YES		NULL	
rrsig20_type	varchar(20)	YES		NULL	
rrsig20_covered_type	varchar(20)	YES		NULL	
rrsig20_algorithm	int(11)	YES		NULL	
rrsig20_label	int(11)	YES		NULL	
rrsig20_original_ttl	decimal(10,0)	YES		NULL	
rrsig20_expiration_time	bigint(20)	YES		NULL	
rrsig20_inception_time	bigint(20)	YES		NULL	

rrsig20_key_tag	int(11)	YES		NULL	
rrsig20_signers_name	varchar(100)	YES		NULL	
rrsig20_signatur	longtext	YES		NULL	
more_rrsig	enum('yes','no','1')	YES		NULL	
ds1	longtext	YES		NULL	
ds1_zone_name	varchar(100)	YES		NULL	
ds1_ttl	int(11)	YES		NULL	
ds1_klasse	varchar(20)	YES		NULL	
ds1_type	varchar(20)	YES		NULL	
ds1_key_tag	int(11)	YES		NULL	
ds1_algorithm	int(11)	YES		NULL	
ds1_digest_type	int(11)	YES		NULL	
ds1_digest	longtext	YES		NULL	
ds2	longtext	YES		NULL	
ds2_zone_name	varchar(100)	YES		NULL	
ds2_ttl	int(11)	YES		NULL	
ds2_klasse	varchar(20)	YES		NULL	
ds2_type	varchar(20)	YES		NULL	
ds2_key_tag	int(11)	YES		NULL	
ds2_algorithm	int(11)	YES		NULL	
ds2_digest_type	int(11)	YES		NULL	
ds2_digest	longtext	YES		NULL	
ds3	longtext	YES		NULL	
ds3_zone_name	varchar(100)	YES		NULL	
ds3_ttl	int(11)	YES		NULL	
ds3_klasse	varchar(20)	YES		NULL	
ds3_type	varchar(20)	YES		NULL	
ds3_key_tag	int(11)	YES		NULL	
ds3_algorithm	int(11)	YES		NULL	
ds3_digest_type	int(11)	YES		NULL	
ds3_digest	longtext	YES		NULL	
ds4	longtext	YES		NULL	
ds4_zone_name	varchar(100)	YES		NULL	
ds4_ttl	int(11)	YES		NULL	
ds4_klasse	varchar(20)	YES		NULL	
ds4_type	varchar(20)	YES		NULL	
ds4_key_tag	int(11)	YES		NULL	
ds4_algorithm	int(11)	YES		NULL	
ds4_digest_type	int(11)	YES		NULL	
ds4_digest	longtext	YES		NULL	
more_ds	enum('yes','no','1')	YES		NULL	
dnskey1	longtext	YES		NULL	
dnskey1_zone_name	varchar(100)	YES		NULL	
dnskey1_ttl	decimal(10,0)	YES		NULL	
dnskey1_klasse	varchar(20)	YES		NULL	
dnskey1_type	varchar(20)	YES		NULL	
dnskey1_flags	int(11)	YES		NULL	
dnskey1_protokoll	int(11)	YES		NULL	
dnskey1_algorithm	int(11)	YES		NULL	
dnskey1_public_key	longtext	YES		NULL	
dnskey1_id	int(11)	YES		NULL	
dnskey1_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey1_key_size	varchar(20)	YES		NULL	
dnskey2	longtext	YES		NULL	
dnskey2_zone_name	varchar(100)	YES		NULL	
dnskey2_ttl	decimal(10,0)	YES		NULL	
dnskey2_klasse	varchar(20)	YES		NULL	
dnskey2_type	varchar(20)	YES		NULL	
dnskey2_flags	int(11)	YES		NULL	
dnskey2_protokoll	int(11)	YES		NULL	
dnskey2_algorithm	int(11)	YES		NULL	
dnskey2_public_key	longtext	YES		NULL	
dnskey2_id	int(11)	YES		NULL	
dnskey2_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey2_key_size	varchar(20)	YES		NULL	
dnskey3	longtext	YES		NULL	
dnskey3_zone_name	varchar(100)	YES		NULL	
dnskey3_ttl	decimal(10,0)	YES		NULL	
dnskey3_klasse	varchar(20)	YES		NULL	
dnskey3_type	varchar(20)	YES		NULL	
dnskey3_flags	int(11)	YES		NULL	

dnskey3_protokoll	int(11)	YES		NULL	
dnskey3_algorithm	int(11)	YES		NULL	
dnskey3_public_key	longtext	YES		NULL	
dnskey3_id	int(11)	YES		NULL	
dnskey3_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey3_key_size	varchar(20)	YES		NULL	
dnskey4	longtext	YES		NULL	
dnskey4_zone_name	varchar(100)	YES		NULL	
dnskey4_ttl	decimal(10,0)	YES		NULL	
dnskey4_klasse	varchar(20)	YES		NULL	
dnskey4_type	varchar(20)	YES		NULL	
dnskey4_flags	int(11)	YES		NULL	
dnskey4_protokoll	int(11)	YES		NULL	
dnskey4_algorithm	int(11)	YES		NULL	
dnskey4_public_key	longtext	YES		NULL	
dnskey4_id	int(11)	YES		NULL	
dnskey4_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey4_key_size	varchar(20)	YES		NULL	
dnskey5	longtext	YES		NULL	
dnskey5_zone_name	varchar(100)	YES		NULL	
dnskey5_ttl	decimal(10,0)	YES		NULL	
dnskey5_klasse	varchar(20)	YES		NULL	
dnskey5_type	varchar(20)	YES		NULL	
dnskey5_flags	int(11)	YES		NULL	
dnskey5_protokoll	int(11)	YES		NULL	
dnskey5_algorithm	int(11)	YES		NULL	
dnskey5_public_key	longtext	YES		NULL	
dnskey5_id	int(11)	YES		NULL	
dnskey5_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey5_key_size	varchar(20)	YES		NULL	
dnskey6	longtext	YES		NULL	
dnskey6_zone_name	varchar(100)	YES		NULL	
dnskey6_ttl	decimal(10,0)	YES		NULL	
dnskey6_klasse	varchar(20)	YES		NULL	
dnskey6_type	varchar(20)	YES		NULL	
dnskey6_flags	int(11)	YES		NULL	
dnskey6_protokoll	int(11)	YES		NULL	
dnskey6_algorithm	int(11)	YES		NULL	
dnskey6_public_key	longtext	YES		NULL	
dnskey6_id	int(11)	YES		NULL	
dnskey6_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey6_key_size	varchar(20)	YES		NULL	
more_dnskey	enum('yes','no','1')	YES		NULL	
nsec3_1	longtext	YES		NULL	
nsec3_1_zone_name	varchar(500)	YES		NULL	
nsec3_1_ttl	int(11)	YES		NULL	
nsec3_1_klasse	varchar(20)	YES		NULL	
nsec3_1_type	varchar(20)	YES		NULL	
nsec3_1_hash_algo	int(11)	YES		NULL	
nsec3_1_iteration	int(11)	YES		NULL	
nsec3_1_flags	int(11)	YES		NULL	
nsec3_1_salt	varchar(200)	YES		NULL	
nsec3_1_next_domain	varchar(256)	YES		NULL	
nsec3_1_bitmap	varchar(256)	YES		NULL	
nsec3_2	longtext	YES		NULL	
nsec3_2_zone_name	varchar(500)	YES		NULL	
nsec3_2_ttl	int(11)	YES		NULL	
nsec3_2_klasse	varchar(20)	YES		NULL	
nsec3_2_type	varchar(20)	YES		NULL	
nsec3_2_hash_algo	int(11)	YES		NULL	
nsec3_2_iteration	int(11)	YES		NULL	
nsec3_2_flags	int(11)	YES		NULL	
nsec3_2_salt	varchar(200)	YES		NULL	
nsec3_2_next_domain	varchar(256)	YES		NULL	
nsec3_2_bitmap	varchar(256)	YES		NULL	
nsec3_3	longtext	YES		NULL	
nsec3_3_zone_name	varchar(500)	YES		NULL	
nsec3_3_ttl	int(11)	YES		NULL	
nsec3_3_klasse	varchar(20)	YES		NULL	
nsec3_3_type	varchar(20)	YES		NULL	
nsec3_3_hash_algo	int(11)	YES		NULL	

nsec3_3_iteration	int(11)	YES		NULL	
nsec3_3_flags	int(11)	YES		NULL	
nsec3_3_salt	varchar(200)	YES		NULL	
nsec3_3_next_domain	varchar(256)	YES		NULL	
nsec3_3_bitmap	varchar(256)	YES		NULL	
more_nsec3	enum('yes','no','1')	YES		NULL	
nsec1	longtext	YES		NULL	
nsec1_zone_name	varchar(100)	YES		NULL	
nsec1_ttl	int(11)	YES		NULL	
nsec1_klasse	varchar(20)	YES		NULL	
nsec1_type	varchar(20)	YES		NULL	
nsec1_next_domain	varchar(256)	YES		NULL	
nsec1_bitmap	varchar(256)	YES		NULL	
nsec2	longtext	YES		NULL	
nsec2_zone_name	varchar(100)	YES		NULL	
nsec2_ttl	int(11)	YES		NULL	
nsec2_klasse	varchar(20)	YES		NULL	
nsec2_type	varchar(20)	YES		NULL	
nsec2_next_domain	varchar(256)	YES		NULL	
nsec2_bitmap	varchar(256)	YES		NULL	
more_nsec	enum('yes','no','1')	YES		NULL	
soa	varchar(200)	YES		NULL	
soa_name_of_zone	varchar(100)	YES		NULL	
soa_Klasse	varchar(20)	YES		NULL	
soa_typ	varchar(20)	YES		NULL	
soa_primary	varchar(256)	YES		NULL	
soa_mail	varchar(256)	YES		NULL	
soa_serial	bigint(20)	YES		NULL	
soa_refresh	int(11)	YES		NULL	
soa_retry	int(11)	YES		NULL	
soa_expire	bigint(20)	YES		NULL	
soa_ttl	int(11)	YES		NULL	
more_soa	enum('yes','no','1')	YES		NULL	
DESCRIBEannonceV4_20150509					
Field	Type	Null	Key	Default	Extra
asNumber	int(11)	NO	PRI	NULL	
routeIPv4	int(11)	NO	PRI	NULL	
source	varchar(255)	NO	PRI	NULL	
DESCRIBEannonceV6_20150509					
Field	Type	Null	Key	Default	Extra
asNumber	int(11)	NO	PRI	NULL	
routeIPv6	int(11)	NO	PRI	NULL	
source	varchar(255)	NO	PRI	NULL	
DESCRIBEautoSystem_20150509					
Field	Type	Null	Key	Default	Extra
number	int(11)	NO	PRI	NULL	
pays	varchar(255)	YES		NULL	
source	varchar(255)	NO	PRI	NULL	
DESCRIBEbgpresults_20150509					
Field	Type	Null	Key	Default	Extra
totalAS	int(11)	NO		NULL	
missedAS	int(11)	NO		NULL	
source	varchar(255)	NO		NULL	
DESCRIBEdnssec_validation_20140430					
Field	Type	Null	Key	Default	Extra
domain	varchar(256)	NO	PRI	NULL	
root_time_dnssec	float	YES		NULL	
tld_time_dnssec	float	YES		NULL	
2ld_time_dnssec	float	YES		NULL	
3ld_time_dnssec	float	YES		NULL	
total_time_dnssec	float	YES		NULL	
root_time_dns	float	YES		NULL	
tld_time_dns	float	YES		NULL	
2ld_time_dns	float	YES		NULL	
3ld_time_dns	float	YES		NULL	
total_time_dns	float	YES		NULL	

secure	enum('trusted','selfsig','bogus','untrusted','notexist','1')	NO		NULL	
security_comments	varchar(256)	YES		NULL	
existence_denied	enum('yes','no','1')	YES		NULL	
root_name	varchar(256)	YES		NULL	
mysql>DESCRIBEdns_20150525					
Field	Type	Null	Key	Default	Extra
fqdn	varchar(512)	NO	PRI	NULL	
domain	varchar(100)	NO	PRI	NULL	
ip	varchar(39)	NO	PRI		
ip_version	enum('4','6')	YES	MUL	NULL	
type	enum('ns','a','mx','txt')	NO	PRI	ns	
mysql>DESCRIBEdns_discover_20150406					
Field	Type	Null	Key	Default	Extra
fqdn	varchar(256)	NO	PRI	NULL	
domain	varchar(100)	NO	PRI	NULL	
ip	varchar(39)	NO	PRI	NULL	
authoritative_udp	enum('error','yes','no')	NO		NULL	
authoritative_tcp	enum('error','yes','no')	NO		NULL	
recursive_udp	enum('error','yes','no')	NO		NULL	
recursive_tcp	enum('error','yes','no')	NO		NULL	
authority	varchar(256)	NO		NULL	
email_address	varchar(256)	NO		NULL	
serial	decimal(10,0)	NO		NULL	
refresh	decimal(10,0)	NO		NULL	
retry	decimal(10,0)	NO		NULL	
expire	decimal(10,0)	NO		NULL	
ttl	decimal(10,0)	NO		NULL	
zone_transfer	enum('allowed','notallowed')	NO		NULL	
zone_transfer_values	blob	YES		NULL	
version_udp	varchar(512)	NO		NULL	
version_tcp	varchar(512)	NO		NULL	
hostname_udp	varchar(512)	NO		NULL	
hostname_tcp	varchar(512)	NO		NULL	
wildcard_udp	enum('error','yes','no')	NO		NULL	
wildcard_tcp	enum('error','yes','no')	NO		NULL	
glue_record_validity	enum('valid','warning','notauthoritative','error')	NO		NULL	
glue_record_info	blob	YES		NULL	
mysql>DESCRIBElookingGlass_20130208					
Field	Type	Null	Key	Default	Extra
url	varchar(255)	NO	PRI	NULL	
username	varchar(255)	YES		NULL	
password	varchar(255)	YES		NULL	
asNumber	int(11)	NO		NULL	
prompt	varchar(255)	NO		NULL	
nb_max_connection	int(11)	NO		NULL	
port	varchar(255)	NO		0	
timeout	int(11)	NO		120	
enable	int(11)	NO		0	
mysql>DESCRIBEnpa_20110224					
Field	Type	Null	Key	Default	Extra
id	int(10)	NO	PRI	NULL	auto_increment
npa	int(11)	YES	MUL	NULL	
city	varchar(100)	YES		NULL	
canton	varchar(10)	YES	MUL	NULL	
mysql>DESCRIBEnvd_cvss_20150406					
Field	Type	Null	Key	Default	Extra
cve	varchar(30)	NO	PRI		
score	decimal(3,1)	YES	MUL	NULL	
access_vector	varchar(16)	YES		NULL	
access_complexity	varchar(16)	YES		NULL	
authentication	varchar(16)	YES		NULL	
confidentiality_impact	varchar(16)	YES		NULL	
integrity_impact	varchar(16)	YES		NULL	
availability_impact	varchar(16)	YES		NULL	
mysql>DESCRIBEnvd_vulns_20150406					

Field	Type	Null	Key	Default	Extra
cve	varchar(30)	YES	MUL	NULL	
vendor	varchar(128)	YES	MUL	NULL	
product	varchar(128)	YES	MUL	NULL	
version	varchar(128)	YES	MUL	NULL	
update	varchar(128)	YES		NULL	
edition	varchar(128)	YES		NULL	
part	varchar(128)	YES		NULL	
mysql>DESCRIBEreverse_dns_20150518					
Field	Type	Null	Key	Default	Extra
ip_range	varchar(50)	NO	MUL	NULL	
ip	varchar(39)	NO		NULL	
domain	varchar(100)	NO	MUL	NULL	
fqdn	varchar(256)	NO		NULL	
mysql>DESCRIBEripe_20150519					
Field	Type	Null	Key	Default	Extra
ip_range	varchar(50)	NO	PRI	NULL	
as_number	varchar(10)	YES	MUL	NULL	
as_name	varchar(50)	YES	MUL	NULL	
mysql>DESCRIBEroot_20120430					
Field	Type	Null	Key	Default	Extra
root_name	varchar(256)	NO	PRI	NULL	
ipv4	varchar(50)	YES		NULL	
ipv6	varchar(50)	YES		NULL	
rrsig1	longtext	YES		NULL	
rrsig1_name_assigned	varchar(100)	YES		NULL	
rrsig1_ttl	decimal(10,0)	YES		NULL	
rrsig1_klasse	varchar(20)	YES		NULL	
rrsig1_type	varchar(20)	YES		NULL	
rrsig1_covered_type	varchar(20)	YES		NULL	
rrsig1_algorithm	int(11)	YES		NULL	
rrsig1_label	int(11)	YES		NULL	
rrsig1_original_ttl	decimal(10,0)	YES		NULL	
rrsig1_expiration_time	bigint(20)	YES		NULL	
rrsig1_inception_time	bigint(20)	YES		NULL	
rrsig1_key_tag	int(11)	YES		NULL	
rrsig1_signers_name	varchar(100)	YES		NULL	
rrsig1_signatur	longtext	YES		NULL	
rrsig2	longtext	YES		NULL	
rrsig2_name_assigned	varchar(100)	YES		NULL	
rrsig2_ttl	decimal(10,0)	YES		NULL	
rrsig2_klasse	varchar(20)	YES		NULL	
rrsig2_type	varchar(20)	YES		NULL	
rrsig2_covered_type	varchar(20)	YES		NULL	
rrsig2_algorithm	int(11)	YES		NULL	
rrsig2_label	int(11)	YES		NULL	
rrsig2_original_ttl	decimal(10,0)	YES		NULL	
rrsig2_expiration_time	bigint(20)	YES		NULL	
rrsig2_inception_time	bigint(20)	YES		NULL	
rrsig2_key_tag	int(11)	YES		NULL	
rrsig2_signers_name	varchar(100)	YES		NULL	
rrsig2_signatur	longtext	YES		NULL	
rrsig3	longtext	YES		NULL	
rrsig3_name_assigned	varchar(100)	YES		NULL	
rrsig3_ttl	decimal(10,0)	YES		NULL	
rrsig3_klasse	varchar(20)	YES		NULL	
rrsig3_type	varchar(20)	YES		NULL	
rrsig3_covered_type	varchar(20)	YES		NULL	
rrsig3_algorithm	int(11)	YES		NULL	
rrsig3_label	int(11)	YES		NULL	
rrsig3_original_ttl	decimal(10,0)	YES		NULL	
rrsig3_expiration_time	bigint(20)	YES		NULL	
rrsig3_inception_time	bigint(20)	YES		NULL	
rrsig3_key_tag	int(11)	YES		NULL	
rrsig3_signers_name	varchar(100)	YES		NULL	
rrsig3_signatur	longtext	YES		NULL	
rrsig4	longtext	YES		NULL	
rrsig4_name_assigned	varchar(100)	YES		NULL	

rrsig4_ttl	decimal(10,0)	YES		NULL	
rrsig4_klasse	varchar(20)	YES		NULL	
rrsig4_type	varchar(20)	YES		NULL	
rrsig4_covered_type	varchar(20)	YES		NULL	
rrsig4_algorithm	int(11)	YES		NULL	
rrsig4_label	int(11)	YES		NULL	
rrsig4_original_ttl	decimal(10,0)	YES		NULL	
rrsig4_expiration_time	bigint(20)	YES		NULL	
rrsig4_inception_time	bigint(20)	YES		NULL	
rrsig4_key_tag	int(11)	YES		NULL	
rrsig4_signers_name	varchar(100)	YES		NULL	
rrsig4_signatur	longtext	YES		NULL	
rrsig5	longtext	YES		NULL	
rrsig5_name_assigned	varchar(100)	YES		NULL	
rrsig5_ttl	decimal(10,0)	YES		NULL	
rrsig5_klasse	varchar(20)	YES		NULL	
rrsig5_type	varchar(20)	YES		NULL	
rrsig5_covered_type	varchar(20)	YES		NULL	
rrsig5_algorithm	int(11)	YES		NULL	
rrsig5_label	int(11)	YES		NULL	
rrsig5_original_ttl	decimal(10,0)	YES		NULL	
rrsig5_expiration_time	bigint(20)	YES		NULL	
rrsig5_inception_time	bigint(20)	YES		NULL	
rrsig5_key_tag	int(11)	YES		NULL	
rrsig5_signers_name	varchar(100)	YES		NULL	
rrsig5_signatur	longtext	YES		NULL	
rrsig6	longtext	YES		NULL	
rrsig6_name_assigned	varchar(100)	YES		NULL	
rrsig6_ttl	decimal(10,0)	YES		NULL	
rrsig6_klasse	varchar(20)	YES		NULL	
rrsig6_type	varchar(20)	YES		NULL	
rrsig6_covered_type	varchar(20)	YES		NULL	
rrsig6_algorithm	int(11)	YES		NULL	
rrsig6_label	int(11)	YES		NULL	
rrsig6_original_ttl	decimal(10,0)	YES		NULL	
rrsig6_expiration_time	bigint(20)	YES		NULL	
rrsig6_inception_time	bigint(20)	YES		NULL	
rrsig6_key_tag	int(11)	YES		NULL	
rrsig6_signers_name	varchar(100)	YES		NULL	
rrsig6_signatur	longtext	YES		NULL	
rrsig7	longtext	YES		NULL	
rrsig7_name_assigned	varchar(100)	YES		NULL	
rrsig7_ttl	decimal(10,0)	YES		NULL	
rrsig7_klasse	varchar(20)	YES		NULL	
rrsig7_type	varchar(20)	YES		NULL	
rrsig7_covered_type	varchar(20)	YES		NULL	
rrsig7_algorithm	int(11)	YES		NULL	
rrsig7_label	int(11)	YES		NULL	
rrsig7_original_ttl	decimal(10,0)	YES		NULL	
rrsig7_expiration_time	bigint(20)	YES		NULL	
rrsig7_inception_time	bigint(20)	YES		NULL	
rrsig7_key_tag	int(11)	YES		NULL	
rrsig7_signers_name	varchar(100)	YES		NULL	
rrsig7_signatur	longtext	YES		NULL	
rrsig8	longtext	YES		NULL	
rrsig8_name_assigned	varchar(100)	YES		NULL	
rrsig8_ttl	decimal(10,0)	YES		NULL	
rrsig8_klasse	varchar(20)	YES		NULL	
rrsig8_type	varchar(20)	YES		NULL	
rrsig8_covered_type	varchar(20)	YES		NULL	
rrsig8_algorithm	int(11)	YES		NULL	
rrsig8_label	int(11)	YES		NULL	
rrsig8_original_ttl	decimal(10,0)	YES		NULL	
rrsig8_expiration_time	bigint(20)	YES		NULL	
rrsig8_inception_time	bigint(20)	YES		NULL	
rrsig8_key_tag	int(11)	YES		NULL	
rrsig8_signers_name	varchar(100)	YES		NULL	
rrsig8_signatur	longtext	YES		NULL	
rrsig9	longtext	YES		NULL	
rrsig9_name_assigned	varchar(100)	YES		NULL	
rrsig9_ttl	decimal(10,0)	YES		NULL	

rrsig9_klasse	varchar(20)	YES		NULL	
rrsig9_type	varchar(20)	YES		NULL	
rrsig9_covered_type	varchar(20)	YES		NULL	
rrsig9_algorithm	int(11)	YES		NULL	
rrsig9_label	int(11)	YES		NULL	
rrsig9_original_ttl	decimal(10,0)	YES		NULL	
rrsig9_expiration_time	bigint(20)	YES		NULL	
rrsig9_inception_time	bigint(20)	YES		NULL	
rrsig9_key_tag	int(11)	YES		NULL	
rrsig9_signers_name	varchar(100)	YES		NULL	
rrsig9_signatur	longtext	YES		NULL	
rrsig10	longtext	YES		NULL	
rrsig10_name_assigned	varchar(100)	YES		NULL	
rrsig10_ttl	decimal(10,0)	YES		NULL	
rrsig10_klasse	varchar(20)	YES		NULL	
rrsig10_type	varchar(20)	YES		NULL	
rrsig10_covered_type	varchar(20)	YES		NULL	
rrsig10_algorithm	int(11)	YES		NULL	
rrsig10_label	int(11)	YES		NULL	
rrsig10_original_ttl	decimal(10,0)	YES		NULL	
rrsig10_expiration_time	bigint(20)	YES		NULL	
rrsig10_inception_time	bigint(20)	YES		NULL	
rrsig10_key_tag	int(11)	YES		NULL	
rrsig10_signers_name	varchar(100)	YES		NULL	
rrsig10_signatur	longtext	YES		NULL	
rrsig11	longtext	YES		NULL	
rrsig11_name_assigned	varchar(100)	YES		NULL	
rrsig11_ttl	decimal(10,0)	YES		NULL	
rrsig11_klasse	varchar(20)	YES		NULL	
rrsig11_type	varchar(20)	YES		NULL	
rrsig11_covered_type	varchar(20)	YES		NULL	
rrsig11_algorithm	int(11)	YES		NULL	
rrsig11_label	int(11)	YES		NULL	
rrsig11_original_ttl	decimal(10,0)	YES		NULL	
rrsig11_expiration_time	bigint(20)	YES		NULL	
rrsig11_inception_time	bigint(20)	YES		NULL	
rrsig11_key_tag	int(11)	YES		NULL	
rrsig11_signers_name	varchar(100)	YES		NULL	
rrsig11_signatur	longtext	YES		NULL	
rrsig12	longtext	YES		NULL	
rrsig12_name_assigned	varchar(100)	YES		NULL	
rrsig12_ttl	decimal(10,0)	YES		NULL	
rrsig12_klasse	varchar(20)	YES		NULL	
rrsig12_type	varchar(20)	YES		NULL	
rrsig12_covered_type	varchar(20)	YES		NULL	
rrsig12_algorithm	int(11)	YES		NULL	
rrsig12_label	int(11)	YES		NULL	
rrsig12_original_ttl	decimal(10,0)	YES		NULL	
rrsig12_expiration_time	bigint(20)	YES		NULL	
rrsig12_inception_time	bigint(20)	YES		NULL	
rrsig12_key_tag	int(11)	YES		NULL	
rrsig12_signers_name	varchar(100)	YES		NULL	
rrsig12_signatur	longtext	YES		NULL	
rrsig13	longtext	YES		NULL	
rrsig13_name_assigned	varchar(100)	YES		NULL	
rrsig13_ttl	decimal(10,0)	YES		NULL	
rrsig13_klasse	varchar(20)	YES		NULL	
rrsig13_type	varchar(20)	YES		NULL	
rrsig13_covered_type	varchar(20)	YES		NULL	
rrsig13_algorithm	int(11)	YES		NULL	
rrsig13_label	int(11)	YES		NULL	
rrsig13_original_ttl	decimal(10,0)	YES		NULL	
rrsig13_expiration_time	bigint(20)	YES		NULL	
rrsig13_inception_time	bigint(20)	YES		NULL	
rrsig13_key_tag	int(11)	YES		NULL	
rrsig13_signers_name	varchar(100)	YES		NULL	
rrsig13_signatur	longtext	YES		NULL	
rrsig14	longtext	YES		NULL	
rrsig14_name_assigned	varchar(100)	YES		NULL	
rrsig14_ttl	decimal(10,0)	YES		NULL	
rrsig14_klasse	varchar(20)	YES		NULL	

rrsig14_type	varchar(20)	YES		NULL	
rrsig14_covered_type	varchar(20)	YES		NULL	
rrsig14_algorithm	int(11)	YES		NULL	
rrsig14_label	int(11)	YES		NULL	
rrsig14_original_ttl	decimal(10,0)	YES		NULL	
rrsig14_expiration_time	bigint(20)	YES		NULL	
rrsig14_inception_time	bigint(20)	YES		NULL	
rrsig14_key_tag	int(11)	YES		NULL	
rrsig14_signers_name	varchar(100)	YES		NULL	
rrsig14_signatur	longtext	YES		NULL	
rrsig15	longtext	YES		NULL	
rrsig15_name_assigned	varchar(100)	YES		NULL	
rrsig15_ttl	decimal(10,0)	YES		NULL	
rrsig15_klasse	varchar(20)	YES		NULL	
rrsig15_type	varchar(20)	YES		NULL	
rrsig15_covered_type	varchar(20)	YES		NULL	
rrsig15_algorithm	int(11)	YES		NULL	
rrsig15_label	int(11)	YES		NULL	
rrsig15_original_ttl	decimal(10,0)	YES		NULL	
rrsig15_expiration_time	bigint(20)	YES		NULL	
rrsig15_inception_time	bigint(20)	YES		NULL	
rrsig15_key_tag	int(11)	YES		NULL	
rrsig15_signers_name	varchar(100)	YES		NULL	
rrsig15_signatur	longtext	YES		NULL	
rrsig16	longtext	YES		NULL	
rrsig16_name_assigned	varchar(100)	YES		NULL	
rrsig16_ttl	decimal(10,0)	YES		NULL	
rrsig16_klasse	varchar(20)	YES		NULL	
rrsig16_type	varchar(20)	YES		NULL	
rrsig16_covered_type	varchar(20)	YES		NULL	
rrsig16_algorithm	int(11)	YES		NULL	
rrsig16_label	int(11)	YES		NULL	
rrsig16_original_ttl	decimal(10,0)	YES		NULL	
rrsig16_expiration_time	bigint(20)	YES		NULL	
rrsig16_inception_time	bigint(20)	YES		NULL	
rrsig16_key_tag	int(11)	YES		NULL	
rrsig16_signers_name	varchar(100)	YES		NULL	
rrsig16_signatur	longtext	YES		NULL	
rrsig17	longtext	YES		NULL	
rrsig17_name_assigned	varchar(100)	YES		NULL	
rrsig17_ttl	decimal(10,0)	YES		NULL	
rrsig17_klasse	varchar(20)	YES		NULL	
rrsig17_type	varchar(20)	YES		NULL	
rrsig17_covered_type	varchar(20)	YES		NULL	
rrsig17_algorithm	int(11)	YES		NULL	
rrsig17_label	int(11)	YES		NULL	
rrsig17_original_ttl	decimal(10,0)	YES		NULL	
rrsig17_expiration_time	bigint(20)	YES		NULL	
rrsig17_inception_time	bigint(20)	YES		NULL	
rrsig17_key_tag	int(11)	YES		NULL	
rrsig17_signers_name	varchar(100)	YES		NULL	
rrsig17_signatur	longtext	YES		NULL	
rrsig18	longtext	YES		NULL	
rrsig18_name_assigned	varchar(100)	YES		NULL	
rrsig18_ttl	decimal(10,0)	YES		NULL	
rrsig18_klasse	varchar(20)	YES		NULL	
rrsig18_type	varchar(20)	YES		NULL	
rrsig18_covered_type	varchar(20)	YES		NULL	
rrsig18_algorithm	int(11)	YES		NULL	
rrsig18_label	int(11)	YES		NULL	
rrsig18_original_ttl	decimal(10,0)	YES		NULL	
rrsig18_expiration_time	bigint(20)	YES		NULL	
rrsig18_inception_time	bigint(20)	YES		NULL	
rrsig18_key_tag	int(11)	YES		NULL	
rrsig18_signers_name	varchar(100)	YES		NULL	
rrsig18_signatur	longtext	YES		NULL	
rrsig19	longtext	YES		NULL	
rrsig19_name_assigned	varchar(100)	YES		NULL	
rrsig19_ttl	decimal(10,0)	YES		NULL	
rrsig19_klasse	varchar(20)	YES		NULL	
rrsig19_type	varchar(20)	YES		NULL	

rrsig19_covered_type	varchar(20)	YES		NULL	
rrsig19_algorithm	int(11)	YES		NULL	
rrsig19_label	int(11)	YES		NULL	
rrsig19_original_ttl	decimal(10,0)	YES		NULL	
rrsig19_expiration_time	bigint(20)	YES		NULL	
rrsig19_inception_time	bigint(20)	YES		NULL	
rrsig19_key_tag	int(11)	YES		NULL	
rrsig19_signers_name	varchar(100)	YES		NULL	
rrsig19_signatur	longtext	YES		NULL	
rrsig20	longtext	YES		NULL	
rrsig20_name_assigned	varchar(100)	YES		NULL	
rrsig20_ttl	decimal(10,0)	YES		NULL	
rrsig20_klasse	varchar(20)	YES		NULL	
rrsig20_type	varchar(20)	YES		NULL	
rrsig20_covered_type	varchar(20)	YES		NULL	
rrsig20_algorithm	int(11)	YES		NULL	
rrsig20_label	int(11)	YES		NULL	
rrsig20_original_ttl	decimal(10,0)	YES		NULL	
rrsig20_expiration_time	bigint(20)	YES		NULL	
rrsig20_inception_time	bigint(20)	YES		NULL	
rrsig20_key_tag	int(11)	YES		NULL	
rrsig20_signers_name	varchar(100)	YES		NULL	
rrsig20_signatur	longtext	YES		NULL	
more_rrsig	enum('yes','no','1')	YES		NULL	
dnskey1	longtext	YES		NULL	
dnskey1_zone_name	varchar(100)	YES		NULL	
dnskey1_ttl	decimal(10,0)	YES		NULL	
dnskey1_klasse	varchar(20)	YES		NULL	
dnskey1_type	varchar(20)	YES		NULL	
dnskey1_flags	int(11)	YES		NULL	
dnskey1_protokoll	int(11)	YES		NULL	
dnskey1_algorithm	int(11)	YES		NULL	
dnskey1_public_key	longtext	YES		NULL	
dnskey1_id	int(11)	YES		NULL	
dnskey1_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey1_key_size	varchar(20)	YES		NULL	
dnskey2	longtext	YES		NULL	
dnskey2_zone_name	varchar(100)	YES		NULL	
dnskey2_ttl	decimal(10,0)	YES		NULL	
dnskey2_klasse	varchar(20)	YES		NULL	
dnskey2_type	varchar(20)	YES		NULL	
dnskey2_flags	int(11)	YES		NULL	
dnskey2_protokoll	int(11)	YES		NULL	
dnskey2_algorithm	int(11)	YES		NULL	
dnskey2_public_key	longtext	YES		NULL	
dnskey2_id	int(11)	YES		NULL	
dnskey2_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey2_key_size	varchar(20)	YES		NULL	
dnskey3	longtext	YES		NULL	
dnskey3_zone_name	varchar(100)	YES		NULL	
dnskey3_ttl	decimal(10,0)	YES		NULL	
dnskey3_klasse	varchar(20)	YES		NULL	
dnskey3_type	varchar(20)	YES		NULL	
dnskey3_flags	int(11)	YES		NULL	
dnskey3_protokoll	int(11)	YES		NULL	
dnskey3_algorithm	int(11)	YES		NULL	
dnskey3_public_key	longtext	YES		NULL	
dnskey3_id	int(11)	YES		NULL	
dnskey3_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey3_key_size	varchar(20)	YES		NULL	
dnskey4	longtext	YES		NULL	
dnskey4_zone_name	varchar(100)	YES		NULL	
dnskey4_ttl	decimal(10,0)	YES		NULL	
dnskey4_klasse	varchar(20)	YES		NULL	
dnskey4_type	varchar(20)	YES		NULL	
dnskey4_flags	int(11)	YES		NULL	
dnskey4_protokoll	int(11)	YES		NULL	
dnskey4_algorithm	int(11)	YES		NULL	
dnskey4_public_key	longtext	YES		NULL	
dnskey4_id	int(11)	YES		NULL	
dnskey4_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	

dnskey4_key_size	varchar(20)	YES		NULL	
dnskey5	longtext	YES		NULL	
dnskey5_zone_name	varchar(100)	YES		NULL	
dnskey5_ttl	decimal(10,0)	YES		NULL	
dnskey5_klasse	varchar(20)	YES		NULL	
dnskey5_type	varchar(20)	YES		NULL	
dnskey5_flags	int(11)	YES		NULL	
dnskey5_protokoll	int(11)	YES		NULL	
dnskey5_algorithm	int(11)	YES		NULL	
dnskey5_public_key	longtext	YES		NULL	
dnskey5_id	int(11)	YES		NULL	
dnskey5_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey5_key_size	varchar(20)	YES		NULL	
dnskey6	longtext	YES		NULL	
dnskey6_zone_name	varchar(100)	YES		NULL	
dnskey6_ttl	decimal(10,0)	YES		NULL	
dnskey6_klasse	varchar(20)	YES		NULL	
dnskey6_type	varchar(20)	YES		NULL	
dnskey6_flags	int(11)	YES		NULL	
dnskey6_protokoll	int(11)	YES		NULL	
dnskey6_algorithm	int(11)	YES		NULL	
dnskey6_public_key	longtext	YES		NULL	
dnskey6_id	int(11)	YES		NULL	
dnskey6_ksk_or_zsk	enum('zsk','ksk','1')	YES		NULL	
dnskey6_key_size	varchar(20)	YES		NULL	
more_dnskey	enum('yes','no','1')	YES		NULL	
soa	varchar(200)	YES		NULL	
soa_name_of_zone	varchar(100)	YES		NULL	
soa_Klasse	varchar(20)	YES		NULL	
soa_typ	varchar(20)	YES		NULL	
soa_primary	varchar(256)	YES		NULL	
soa_mail	varchar(256)	YES		NULL	
soa_serial	bigint(20)	YES		NULL	
soa_refresh	int(11)	YES		NULL	
soa_retry	int(11)	YES		NULL	
soa_expire	bigint(20)	YES		NULL	
soa_ttl	int(11)	YES		NULL	
more_soa	enum('yes','no','1')	YES		NULL	
mysql>DESCRIBErouteIPv4_20150509					
Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
prefixe	varchar(255)	NO		NULL	
mysql>DESCRIBErouteIPv6_20150509					
Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
prefixe	varchar(255)	NO		NULL	
mysql>DESCRIBEscan_os_20150501					
Field	Type	Null	Key	Default	Extra
ip	varchar(15)	NO		NULL	
os	varchar(256)	YES		NULL	
accuracy	int(10)	YES		NULL	
mysql>DESCRIBEscan_services_20150501					
Field	Type	Null	Key	Default	Extra
ip	varchar(15)	NO		NULL	
port	int(10)	YES		NULL	
service	varchar(256)	YES		NULL	
software	varchar(256)	YES		NULL	
version	varchar(256)	YES		NULL	
extra_info	varchar(256)	YES		NULL	
mysql>DESCRIBEsmtp_discover_20150413					
Field	Type	Null	Key	Default	Extra
ip	varchar(39)	NO	PRI	NULL	
server_name_p25	varchar(100)	NO		NULL	
server_type_p25	varchar(100)	NO		NULL	
server_version_p25	varchar(100)	NO		NULL	
os_name_p25	varchar(100)	NO		NULL	

helo_p25	enum('null','yes','no')	NO		NULL	
ehlo_p25	enum('null','yes','no')	NO		NULL	
verify_p25	enum('null','yes','no')	NO		NULL	
expn_p25	enum('null','yes','no')	NO		NULL	
atrn_p25	enum('null','yes','no')	NO		NULL	
auth_p25	enum('null','yes','no')	NO		NULL	
bdat_p25	enum('null','yes','no')	NO		NULL	
binary_mime_p25	enum('null','yes','no')	NO		NULL	
checkpoint_p25	enum('null','yes','no')	NO		NULL	
chunking_p25	enum('null','yes','no')	NO		NULL	
deliver_by_p25	enum('null','yes','no')	NO		NULL	
dsn_p25	enum('null','yes','no')	NO		NULL	
enhanced_status_codes_p25	enum('null','yes','no')	NO		NULL	
etrn_p25	enum('null','yes','no')	NO		NULL	
help_p25	enum('null','yes','no')	NO		NULL	
mtrk_p25	enum('null','yes','no')	NO		NULL	
no_soliciting_p25	enum('null','yes','no')	NO		NULL	
onex_p25	enum('null','yes','no')	NO		NULL	
pipelining_p25	enum('null','yes','no')	NO		NULL	
saml_p25	enum('null','yes','no')	NO		NULL	
send_p25	enum('null','yes','no')	NO		NULL	
soml_p25	enum('null','yes','no')	NO		NULL	
size_p25	enum('null','yes','no')	NO		NULL	
start_tls_p25	enum('null','yes','no')	NO		NULL	
turn_p25	enum('null','yes','no')	NO		NULL	
utf8_smtp_p25	enum('null','yes','no')	NO		NULL	
verb_p25	enum('null','yes','no')	NO		NULL	
8bit_mime_p25	enum('null','yes','no')	NO		NULL	
banner_p25	blob	YES		NULL	
help_message_p25	blob	YES		NULL	
server_name_p587	varchar(100)	NO		NULL	
server_type_p587	varchar(100)	NO		NULL	
server_version_p587	varchar(100)	NO		NULL	
os_name_p587	varchar(100)	NO		NULL	
helo_p587	enum('null','yes','no')	NO		NULL	
ehlo_p587	enum('null','yes','no')	NO		NULL	
verify_p587	enum('null','yes','no')	NO		NULL	
expn_p587	enum('null','yes','no')	NO		NULL	
atrn_p587	enum('null','yes','no')	NO		NULL	
auth_p587	enum('null','yes','no')	NO		NULL	
bdat_p587	enum('null','yes','no')	NO		NULL	
binary_mime_p587	enum('null','yes','no')	NO		NULL	
checkpoint_p587	enum('null','yes','no')	NO		NULL	
chunking_p587	enum('null','yes','no')	NO		NULL	
deliver_by_p587	enum('null','yes','no')	NO		NULL	
dsn_p587	enum('null','yes','no')	NO		NULL	
enhanced_status_codes_p587	enum('null','yes','no')	NO		NULL	
etrn_p587	enum('null','yes','no')	NO		NULL	
help_p587	enum('null','yes','no')	NO		NULL	
mtrk_p587	enum('null','yes','no')	NO		NULL	
no_soliciting_p587	enum('null','yes','no')	NO		NULL	
onex_p587	enum('null','yes','no')	NO		NULL	
pipelining_p587	enum('null','yes','no')	NO		NULL	
saml_p587	enum('null','yes','no')	NO		NULL	
send_p587	enum('null','yes','no')	NO		NULL	
soml_p587	enum('null','yes','no')	NO		NULL	
size_p587	enum('null','yes','no')	NO		NULL	
start_tls_p587	enum('null','yes','no')	NO		NULL	
turn_p587	enum('null','yes','no')	NO		NULL	
utf8_smtp_p587	enum('null','yes','no')	NO		NULL	
verb_p587	enum('null','yes','no')	NO		NULL	
8bit_mime_p587	enum('null','yes','no')	NO		NULL	
banner_p587	blob	YES		NULL	
help_message_p587	blob	YES		NULL	
mysql>DESCRIBEssl_discover_20150504					
Field	Type	Null	Key	Default	Extra
fqdn	varchar(256)	NO	PRI	NULL	
domain	varchar(100)	NO	MUL	NULL	
valid	tinyint(1)	NO	MUL	NULL	
wronghost	varchar(256)	YES		NULL	

verif_error	tinyint(1)	NO	MUL	NULL	
cert_fingerprint	varchar(32)	YES	MUL	NULL	
cert	mediumblob	YES		NULL	
not_before	datetime	YES	MUL	NULL	
not_after	datetime	YES	MUL	NULL	
keylength	int(4)	YES	MUL	NULL	
issuer	varchar(256)	YES		NULL	
sigalgo	varchar(7)	YES	MUL	NULL	
ev	tinyint(1)	YES	MUL	NULL	
ciphers	varchar(10)	YES	MUL	NULL	
ssl2	enum('yes','no')	YES	MUL	NULL	
weak_tls1	enum('yes','no')	YES	MUL	NULL	
low_tls1	enum('yes','no')	YES	MUL	NULL	
med_tls1	enum('yes','no')	YES	MUL	NULL	
high_tls1	enum('yes','no')	YES	MUL	NULL	
mysql>DESCRIBEvoisins_20150509					
Field	Type	Null	Key	Default	Extra
as1	int(11)	NO	PRI	NULL	
as2	int(11)	NO	PRI	NULL	
source	varchar(255)	NO	PRI	NULL	
mysql>DESCRIBEweb_fingerprint_20150522					
Field	Type	Null	Key	Default	Extra
fqdn	varchar(256)	NO	PRI	NULL	
domain	varchar(100)	NO	MUL	NULL	
effective_url	varchar(512)	NO	MUL	NULL	
server	varchar(512)	NO	MUL	NULL	
powered_by	varchar(256)	NO	MUL	NULL	
generator	varchar(512)	NO	MUL	NULL	
apps	varchar(512)	NO	MUL	NULL	
mysql>DESCRIBEweb_fingerprint_full_20150522					
Field	Type	Null	Key	Default	Extra
fqdn	varchar(256)	NO		NULL	
file	mediumblob	NO		NULL	
mysql>DESCRIBEwhois_20110201					
Field	Type	Null	Key	Default	Extra
domain	varchar(100)	NO	PRI	NULL	
ace	varchar(100)	YES		NULL	
holder	varchar(500)	YES		NULL	
technical_contact	varchar(500)	YES		NULL	
dns	varchar(500)	YES		NULL	
npa	int(11)	YES	MUL	NULL	

Annexe 6 : Taille des tables			
Table	**Data [GB]**	**Index[MB]**	**Total [MB]**
archive.1ld_20131129	10,66	45,38	10961,22
archive.1ld_20130729	8,75	46,62	9006,62
archive.1ld_20130930	8,75	45,43	9005,43
archive.1ld_20130528	8,72	45,35	8974,63
archive.2ld_20121217	4,86	44,76	5021,4
archive.2ld_20131129	4	45,37	4141,37
archive.2ld_20130930	3,98	45,38	4120,9
archive.2ld_20130729	3,97	46,46	4111,74
archive.2ld_20130830	3,97	45,21	4110,49
archive.2ld_20120929	3,83	44,72	3966,64
archive.2ld_20120430	3,79	44,67	3925,63
archive.2ld_20120331	3,78	44,45	3915,17
archive.2ld_20111212	3,76	44,78	3895,02
archive.2ld_20140430	2,96	42,32	3073,36
archive.2ld_20140331	2,96	42,28	3073,32
archive.ssl_discover_20111128	1,52	141,27	1697,75
archive.ssl_discover_20121210	1,52	137,24	1693,72
archive.ssl_discover_20140210	1,52	131,17	1687,65
archive.ssl_discover_20150504	1,52	130,11	1686,59
archive.ssl_discover_20120423	1,51	139,46	1685,7
archive.ssl_discover_20110822	1,5	144,25	1680,25
archive.ssl_discover_20130603	1,51	133,83	1680,07
archive.ssl_discover_20130701	1,5	132,67	1668,67
archive.ssl_discover_20120305	1,49	138,44	1664,2
archive.ssl_discover_20120702	1,49	137,92	1663,68
archive.ssl_discover_20120806	1,49	137,08	1662,84
archive.ssl_discover_20130304	1,49	134,02	1659,78
archive.ssl_discover_20130805	1,49	131,8	1657,56
archive.ssl_discover_20120507	1,48	137,44	1652,96
archive.ssl_discover_20110530	1,47	140,91	1646,19
archive.ssl_discover_20140106	1,52	41,29	1597,77
archive.dns_discover_20120910	1,09	126,8	1242,96
archive.dns_discover_20130408	1,09	123,7	1239,86
archive.dns_discover_20130513	1,09	123,38	1239,54
archive.dns_discover_20130610	1,09	123,01	1239,17
archive.dns_discover_20130708	1,08	121,63	1227,55
archive.dns_discover_20130311	1,07	119,91	1215,59
archive.dns_discover_20131111	1,06	119,65	1205,09
archive.dns_discover_20131028	1,06	119,65	1205,09
archive.dns_discover_20120813	1,05	125,66	1200,86
archive.dns_discover_20130909	1,05	118,78	1193,98
archive.dns_discover_20140407	1,04	120,67	1185,63
archive.dns_discover_20111031	1,03	122,2	1176,92
archive.dns_discover_20111121	1,03	121,42	1176,14
archive.dns_discover_20110919	1,03	121,3	1176,02
archive.dns_discover_20111205	1,02	120,72	1165,2
archive.dns_discover_20120507	1,02	120,5	1164,98
archive.dns_discover_20110829	1,01	125,9	1160,14

archive.dns_discover_20120116	1	121,78	1145,78
archive.dns_discover_20120409	1	121,28	1145,28
archive.dns_discover_20120326	0,99	119,7	1133,46
archive.dns_discover_20110613	0,97	126,03	1119,31
archive.dns_discover_20110530	0,96	126,49	1109,53
archive.dns_discover_20150406	0,89	117,81	1029,17
archive.dns_discover_20110328	0,85	121,9	992,3
archive.reverse_dns_20150518	0,79	56,46	865,42
archive.ssl_discover_20110321	0,74	28,49	786,25
archive.reverse_dns_20131223	0,7	55,11	771,91
archive.reverse_dns_20130715	0,7	54,99	771,79
archive.reverse_dns_20130617	0,7	54,74	771,54
archive.reverse_dns_20131118	0,69	54,78	761,34
archive.reverse_dns_20121015	0,69	54,33	760,89
archive.reverse_dns_20130128	0,69	53,97	760,53
archive.reverse_dns_20121224	0,69	53,7	760,26
archive.dns_20110704	0,26	474	740,24
archive.reverse_dns_20120720	0,67	52,73	738,81
archive.reverse_dns_20120618	0,67	52,54	738,62
archive.reverse_dns_20130819	0,67	51,84	737,92
archive.dns_20120820	0,26	471,02	737,26
archive.dns_20121119	0,26	465,78	732,02
archive.dns_20110926	0,26	463,57	729,81
archive.dns_20120924	0,26	461,64	727,88
archive.dns_20111024	0,26	460,77	727,01
archive.dns_20111128	0,26	458,94	725,18
archive.dns_20110829	0,26	457,28	723,52
archive.dns_20130325	0,26	457,22	723,46
archive.dns_20111205	0,26	456,92	723,16
archive.dns_20110418	0,25	466,52	722,52
archive.dns_20120123	0,26	455,46	721,7
archive.dns_20121224	0,26	453,38	719,62
archive.dns_20130422	0,25	452,68	708,68
archive.dns_20130520	0,25	451,59	707,59
archive.reverse_dns_20110516	0,64	50,14	705,5
archive.dns_20130121	0,25	449,21	705,21
archive.reverse_dns_20131021	0,64	47,13	702,49
archive.dns_20130624	0,25	446,25	702,25
archive.dns_20130722	0,25	444,27	700,27
archive.dns_20130826	0,24	436,62	682,38
archive.dns_20140224	0,25	422,96	678,96
archive.reverse_dns_20140331	0,61	46,13	670,77
archive.dns_20150525	0,24	421,6	667,36
archive.dns_20140428	0,24	420,97	666,73
archive.dns_20110228	0,21	370,46	585,5
archive.dns_20110328	0,21	367	582,04
archive.1ld_20140430	0,31	2,38	319,82
archive.whois_20110201	0,25	54,17	310,17
archive.web_fingerprint_20110425	0,15	122,58	276,18
archive.web_fingerprint_20110530	0,15	122,46	276,06

archive.web_fingerprint_20110831	0,15	120,26	273,86
archive.web_fingerprint_20110905	0,15	120,14	273,74
archive.web_fingerprint_20110630	0,15	119,02	272,62
archive.web_fingerprint_20110702	0,14	118,79	262,15
archive.web_fingerprint_20121022	0,14	117,35	260,71
archive.web_fingerprint_20120229	0,14	116,21	259,57
archive.web_fingerprint_20120622	0,14	115,78	259,14
archive.web_fingerprint_20120525	0,14	115,03	258,39
archive.reverse_dns_20130916	0,25	0	256
archive.web_fingerprint_20130322	0,13	112,92	246,04
archive.web_fingerprint_20130822	0,13	109,51	242,63
archive.web_fingerprint_20131122	0,12	107,25	230,13
archive.dnssec_validation_20130729	0,15	46,62	200,22
archive.dnssec_validation_20131129	0,15	45,38	198,98
archive.dnssec_validation_20130715	0,15	45,27	198,87
archive.dnssec_validation_20130128	0,15	45,2	198,8
archive.dnssec_validation_20120530	0,15	45,18	198,78
archive.dnssec_validation_20120629	0,15	45,12	198,72
archive.dnssec_validation_20120929	0,15	45,02	198,62
archive.dnssec_validation_20140128	0,15	42,81	196,41
archive.dnssec_validation_20140331	0,15	42,71	196,31
archive.dnssec_validation_20110919	0,15	42,69	196,29
archive.dnssec_validation_20140430	0,15	42,59	196,19
archive.dns_discover_20131202	0,16	23,92	187,76
archive.web_fingerprint_20140107	0,08	104,37	186,29
archive.web_fingerprint_20150522	0,07	98,54	170,22
archive.web_fingerprint_20110330	0,08	75,96	157,88
archive.nvd_vulns_20150406	0,09	20,26	112,42
archive.nvd_vulns_20140324	0,08	18,78	100,7
archive.nvd_vulns_20140210	0,08	18,5	100,42
archive.nvd_vulns_20131216	0,08	18,22	100,14
archive.nvd_vulns_20131007	0,08	17,79	99,71
archive.nvd_vulns_20130902	0,08	17,62	99,54
archive.nvd_vulns_20130701	0,07	17,45	89,13
archive.nvd_vulns_20130304	0,07	16,89	88,57
archive.nvd_vulns_20130107	0,07	16,29	87,97
archive.nvd_vulns_20121015	0,07	15,92	87,6
archive.nvd_vulns_20120903	0,07	15,54	87,22
archive.nvd_vulns_20120827	0,07	15,4	87,08
archive.nvd_vulns_20120702	0,07	15,32	87
archive.nvd_vulns_20120604	0,06	15,1	76,54
archive.nvd_vulns_20120402	0,06	14,28	75,72
archive.nvd_vulns_20120312	0,06	13,72	75,16
archive.nvd_vulns_20110829	0,04	10,27	51,23
archive.nvd_vulns_20110620	0,04	9,16	50,12
archive.nvd_vulns_20110530	0,04	8,83	49,79
archive.nvd_vulns_20110425	0,03	8,17	38,89
archive.nvd_vulns_20110321	0,03	7,92	38,64
archive.scan_services_20110901	0,03	0	30,72
archive.scan_services_20110301	0,03	0	30,72

archive.scan_os_20121001	0,02	0	20,48
archive.scan_services_20120801	0,02	0	20,48
archive.scan_services_20130301	0,02	0	20,48
archive.scan_services_20120701	0,02	0	20,48
archive.scan_os_20120901	0,02	0	20,48
archive.scan_services_20120501	0,02	0	20,48
archive.scan_os_20120701	0,02	0	20,48
archive.scan_services_20120416	0,02	0	20,48
archive.scan_os_20120501	0,02	0	20,48
archive.scan_os_20120416	0,02	0	20,48
archive.scan_services_20120301	0,02	0	20,48
archive.scan_os_20120301	0,02	0	20,48
archive.scan_services_20150501	0,02	0	20,48
archive.scan_services_20140401	0,02	0	20,48
archive.scan_services_20140201	0,02	0	20,48
archive.scan_services_20140101	0,02	0	20,48
archive.scan_services_20131201	0,02	0	20,48
archive.scan_services_20131001	0,02	0	20,48
archive.scan_services_20130901	0,02	0	20,48
archive.scan_os_20130401	0,02	0	20,48
archive.scan_services_20121201	0,02	0	20,48
archive.scan_services_20130501	0,02	0	20,48
archive.scan_os_20130301	0,02	0	20,48
archive.scan_services_20121101	0,02	0	20,48
archive.scan_services_20130401	0,02	0	20,48
archive.scan_services_20120901	0,02	0	20,48
archive.smtp_discover_20120611	0,01	0,99	11,23
archive.smtp_discover_20111017	0,01	0,98	11,22
archive.smtp_discover_20110523	0,01	0,98	11,22
archive.smtp_discover_20110926	0,01	0,97	11,21
archive.smtp_discover_20110808	0,01	0,97	11,21
archive.smtp_discover_20130211	0,01	0,97	11,21
archive.smtp_discover_20121210	0,01	0,97	11,21
archive.smtp_discover_20120116	0,01	0,97	11,21
archive.smtp_discover_20110620	0,01	0,96	11,2
archive.smtp_discover_20110425	0,01	0,96	11,2
archive.smtp_discover_20131111	0,01	0,96	11,2
archive.smtp_discover_20130513	0,01	0,96	11,2
archive.smtp_discover_20130408	0,01	0,96	11,2
archive.smtp_discover_20120507	0,01	0,96	11,2
archive.smtp_discover_20111121	0,01	0,96	11,2
archive.smtp_discover_20130708	0,01	0,95	11,19
archive.smtp_discover_20130107	0,01	0,95	11,19
archive.smtp_discover_20140210	0,01	0,94	11,18
archive.smtp_discover_20130610	0,01	0,94	11,18
archive.smtp_discover_20150413	0,01	0,92	11,16
archive.scan_services_20110501	0,01	0	10,24
archive.scan_os_20150501	0,01	0	10,24
archive.scan_os_20140401	0,01	0	10,24
archive.scan_os_20140201	0,01	0	10,24

archive.scan_os_20140101	0,01	0	10,24
archive.scan_os_20131101	0,01	0	10,24
archive.scan_os_20131001	0,01	0	10,24
archive.annonceV4_20130509	0	6,03	6,03
archive.annonceV4_20130219	0	6,03	6,03
archive.annonceV4_20130809	0	5,03	5,03
archive.annonceV4_20150509	0	4,03	4,03
archive.annonceV4_20140509	0	4,03	4,03
archive.annonceV4_20140409	0	4,03	4,03
archive.annonceV4_20131209	0	4,03	4,03
archive.annonceV4_20131009	0	4,03	4,03
archive.annonceV4_20130609	0	4,03	4,03
archive.voisins_20131109	0	1,52	1,52
archive.voisins_20131009	0	1,52	1,52
archive.voisins_20130909	0	1,52	1,52
archive.voisins_20130509	0	1,52	1,52
archive.voisins_20130409	0	1,52	1,52
archive.voisins_20130219	0	1,52	1,52
archive.voisins_20150509	0	1,52	1,52
archive.voisins_20131209	0	1,52	1,52
archive.voisins_20140309	0	1,52	1,52
archive.ripe_20130319	0	1,3	1,3
archive.ripe_20130119	0	1,3	1,3
archive.ripe_20121219	0	1,29	1,29
archive.ripe_20150519	0	1,27	1,27
archive.ripe_20140419	0	1,25	1,25
archive.ripe_20140319	0	1,25	1,25
archive.ripe_20140119	0	1,24	1,24
archive.ripe_20131204	0	1,24	1,24
archive.ripe_20120819	0	1,24	1,24
archive.ripe_20120719	0	1,24	1,24
archive.ripe_20140219	0	1,24	1,24
archive.ripe_20131019	0	1,23	1,23
archive.ripe_20130919	0	1,23	1,23
archive.ripe_20120521	0	1,23	1,23
archive.ripe_20120313	0	1,22	1,22
archive.ripe_20120319	0	1,21	1,21
archive.ripe_20110808	0	1,19	1,19
archive.ripe_20110804	0	1,19	1,19
archive.ripe_20110825	0	1,19	1,19
archive.ripe_20110901	0	1,19	1,19
archive.ripe_20110824	0	1,19	1,19
archive.ripe_20110820	0	1,19	1,19
archive.ripe_20110817	0	1,19	1,19
archive.nvd_cvss_20150406	0	1,18	1,18
archive.ripe_20110515	0	1,17	1,17
archive.ripe_20110714	0	1,17	1,17
archive.ripe_20110712	0	1,17	1,17
archive.ripe_20110710	0	1,17	1,17
archive.ripe_20110707	0	1,17	1,17

archive.ripe_20110704	0	1,17	1,17
archive.ripe_20110626	0	1,17	1,17
archive.ripe_20110624	0	1,17	1,17
archive.ripe_20110614	0	1,17	1,17
archive.ripe_20110621	0	1,17	1,17
archive.ripe_20110611	0	1,17	1,17
archive.ripe_20110605	0	1,17	1,17
archive.ripe_20110520	0	1,16	1,16
archive.ripe_20110413	0	1,16	1,16
archive.ripe_20110406	0	1,16	1,16
archive.ripe_20110403	0	1,16	1,16
archive.ripe_20110513	0	1,16	1,16
archive.ripe_20110330	0	1,16	1,16
archive.ripe_20110509	0	1,16	1,16
archive.ripe_20110508	0	1,16	1,16
archive.ripe_20110506	0	1,16	1,16
archive.ripe_20110503	0	1,16	1,16
archive.ripe_20110530	0	1,16	1,16
archive.ripe_20110427	0	1,16	1,16
archive.ripe_20110527	0	1,16	1,16
archive.ripe_20110423	0	1,16	1,16
archive.ripe_20110525	0	1,16	1,16
archive.ripe_20110523	0	1,16	1,16
archive.ripe_20110421	0	1,16	1,16
archive.ripe_20110325	0	1,14	1,14
archive.ripe_20110322	0	1,14	1,14
archive.ripe_20110315	0	1,14	1,14
archive.nvd_cvss_20140210	0	1,03	1,03
archive.nvd_cvss_20131216	0	1,01	1,01
archive.nvd_cvss_20110509	0	1	1
archive.nvd_cvss_20130902	0	0,98	0,98
archive.nvd_cvss_20130805	0	0,97	0,97
archive.nvd_cvss_20130603	0	0,96	0,96
archive.nvd_cvss_20130401	0	0,95	0,95
archive.nvd_cvss_20130304	0	0,94	0,94
archive.nvd_cvss_20130204	0	0,93	0,93
archive.nvd_cvss_20130107	0	0,92	0,92
archive.nvd_cvss_20121015	0	0,91	0,91
archive.nvd_cvss_20120903	0	0,89	0,89
archive.nvd_cvss_20120702	0	0,87	0,87
archive.nvd_cvss_20120507	0	0,86	0,86
archive.nvd_cvss_20120402	0	0,85	0,85
archive.nvd_cvss_20120312	0	0,85	0,85
archive.nvd_cvss_20110905	0	0,81	0,81
archive.nvd_cvss_20110627	0	0,8	0,8
archive.nvd_cvss_20110801	0	0,8	0,8
archive.nvd_cvss_20110530	0	0,79	0,79
archive.nvd_cvss_20110404	0	0,78	0,78
archive.annonceV6_20150509	0	0,25	0,25
archive.annonceV6_20130509	0	0,22	0,22

archive.annonceV6_20140409	0	0,19	0,19
archive.annonceV6_20130319	0	0,16	0,16
archive.annonceV6_20130609	0	0,13	0,13
archive.npa_20110224	0	0,12	0,12
archive.django_flatpage_sites	0	0,05	0,05
archive.auth_user_user_permissions	0	0,05	0,05
archive.auth_user_groups	0	0,05	0,05
archive.auth_group_permissions	0	0,05	0,05
archive.django_admin_log	0	0,03	0,03
archive.auth_permission	0	0,03	0,03
archive.django_session	0	0,02	0,02
archive.registration_registrationprofile	0	0,02	0,02
archive.django_flatpage	0	0,02	0,02
archive.django_content_type	0	0,02	0,02
archive.cvalidationstate_clientvalidationstate	0	0,02	0,02
archive.captcha_captchastore	0	0,02	0,02
archive.auth_user	0	0,02	0,02
archive.auth_message	0	0,02	0,02
archive.auth_group	0	0,02	0,02
archive.root_20120430	0	0,01	0,01
archive.root_20110919	0	0,01	0,01
archive.3ld_20140430	0	0	0
information_schema.TRIGGERS	0	0	0
information_schema.EVENTS	0	0	0
archive.3ld_20140331	0	0	0
archive.3ld_20140128	0	0	0
archive.3ld_20130930	0	0	0
information_schema.COLUMNS	0	0	0
archive.3ld_20130830	0	0	0
archive.3ld_20130729	0	0	0
archive.3ld_20130319	0	0	0
archive.3ld_20130128	0	0	0
archive.blacklist_20140114	0	0	0
archive.blacklist_20110316	0	0	0
archive.3ld_20121217	0	0	0
archive.3ld_20120729	0	0	0
information_schema.ROUTINES	0	0	0
archive.3ld_20120629	0	0	0
information_schema.PLUGINS	0	0	0
archive.3ld_20120430	0	0	0
information_schema.PARTITIONS	0	0	0
archive.3ld_20111212	0	0	0
information_schema.PARAMETERS	0	0	0
information_schema.PROCESSLIST	0	0	0
archive.scan_os_20130101	0	0	0
information_schema.VIEWS	0	0	0

Annexes : Documentation Finale

6. Tables SQL

Ce chapitre critique les résultats de l'annexe 5 et recense les requêtes utilisées afin de récupérer les différentes informations sur les tables MySQL.

6.1. Remarques sur la taille des anciennes tables

On remarque que certaines tables plus anciennes (1ld, 2ld) ont des tailles très importantes par rapport à la dernière version de la table.

Les scripts de récoltes de ces données ne sont pas très fiables et sont encore en période d'essai. La dernière version de la table 2ld représente environ 3GB tandis que celle de 1ld est de 320 MB.

On remarque une évolution plus proche des dernières versions qui tendent vers la taille actuelle.

Néanmoins, ces scripts contenant encore quelques bugs, ils ne sont pas exécutés encore régulièrement (dernière exécution en 2014). Il est donc difficile d'estimer si la taille des données récoltées est correcte.

6.2. Taille des indexes et des tables

Pour obtenir la taille des champs donnée et index, on utilise la requête SQL suivante :

```
SELECT concat(table_schema,'.',table_name) tables,
concat(round(data_length/(1024*1024),2),'M') data_size,
concat(round(index_length/(1024*1024),2),'M') index_size
FROM information_schema.TABLES
WHERE index_length > 0
ORDER BY index_size DESC;
```

6.3. Structure d'une table

DESCRIBE table_name

7. Map/Reduce

Ce chapitre contient quelques informations supplémentaires afin de programmer un MapReduce.

7.1. Programmer un Map/Reduce

Chacune des parties d'un Map/Reduce peut être écrite par le développeur, mais au minimum le Mapper, le Reducer et le code du driver.

Pour lancer l'exemple « wordcount », il suffit d'utiliser la commande suivante :

```
hadoop jar hadoop-*-examples.jar wordcount [-m <#maps>] [-r <#reducers>] <in-dir> <out-dir>
```

Tous les fichiers du dossier « in-dir » seront lus et le nombre de mots seront écrits dans le dossier « out-dir ».

Le code ci-dessous représente un compteur de mots en Java :

```
package org.myorg;

import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();

   public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
         word.set(tokenizer.nextToken());
         context.write(word, one);
      }
   }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

   public void reduce(Text key, Iterable<IntWritable> values, Context context)
     throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
         sum += val.get();
      }
      context.write(key, new IntWritable(sum));
   }
 }

 public static void main(String[] args) throws Exception {
   Configuration conf = new Configuration();

      Job job = new Job(conf, "wordcount");

   job.setOutputKeyClass(Text.class);
   job.setOutputValueClass(IntWritable.class);

   job.setMapperClass(Map.class);
   job.setReducerClass(Reduce.class);

   job.setInputFormatClass(TextInputFormat.class);
   job.setOutputFormatClass(TextOutputFormat.class);

   FileInputFormat.addInputPath(job, new Path(args[0]));
   FileOutputFormat.setOutputPath(job, new Path(args[1]));

   job.waitForCompletion(true);
 }
}
```

Chaque Job Map/Reduce est composé de trois parties :

- Le driver code
  - Le code exécuté sur le client et qui lance le Job.
- Le Mapper
- Le Reducer

7.2. Lire l'entrée du Mapper depuis HDFS

Les données passées au Mapper sont spécifiées par un format d'entrée dans le driver code. Il définit la location des données entrées et comment les séparer pour le traitement.

Chaque Mapper travaille avec une seule part des données. Le format d'entrée est défini par les objets `RecordReader` à extraire (paires clé, valeur) depuis la source *input*.

`FilelnputFormat` est la classe de base utilisée pour tous les formats d'entrée de type fichier. Chaque ligne terminée par un \n est traitée comme une valeur. La clé est l'offset (en byte) depuis le début du fichier. La classe `KeyValueTextlnputFormat` map les lignes terminées par \n comme des 'key SEP value'. Par défaut, le séparateur (SEP) est une tabluation. La classe `SequenceFilelnputFormat` est un fichier binaire de paires clé/valeur avec des métadonnées additionnelles. La classe `SequenceFileAsTextlnputFormat` est similaire, mais map une combinaison (`key.toString( ), value.toString( )`).

Les clés et les valeurs dans Hadoop sont des objets. Les valeurs sont des objets implémentant l'interface `Writable`. Les clés implémentent l'interface `writableComparable`.

Writable

Hadoop définit ses propres types de variables. Ainsi, il faudra utiliser les types suivants :

- `IntWritable` pour les ints
- `LongWritable` pour les longs
- `FloatWritable` pour les floats
- `DoubleWritable` pour les doubles
- `Text` pour les strings
- Etc.

L'interface `Writables` rend la sérialisation rapides et faciles pour Hadoop. Chaque type de valeur doit l'implémenter.

WritableComparable

Un objet `WritableComparable` est un objet `Writable` qui peut également être comparable. Ainsi, deux objets `WritableComparable` peuvent être comparés pour déterminer leur ordre. Les clés doivent être de type `WritableComparable` car elles doivent être passées au Reducer de manière ordonnées.

À noter que tous les types `Writable` de Hadoop implémentent les interfaces `Writable` et `WritableComparable`.

Driver

Le code du driver se lance sur la machine du client. Il configure le Job et l'envoie au cluster.

N°	Description	Attendu	Test	Remarques
100	**Importation des données**			
101	Importation table MySQL NetObservatory --> Hadoop local avec Sqoop	Un fichier CSV du contenu de la table dans un dossier comportant le nom de la table	OK	
102	Importation table MySQL NetObservatory (BD05) --> Hadoop DAPLAB avec Sqoop	Un fichier CSV du contenu de la table dans un dossier comportant le nom de la table	Partial	BD05 n'a plus accès à pubgw01 (Firewall) mais a fonctionné au début du projet
103	Importation table MySQL Réseau EIA --> Hadoop local avec Sqoop	Un fichier CSV du contenu de la table dans un dossier comportant le nom de la table	OK	
104	Importation table MySQL Réseau EIA --> Hadoop DAPLAB avec Sqoop	Un fichier CSV du contenu de la table dans un dossier comportant le nom de la table	OK	
105	Importation table MySQL Backup sql.gz NetObservatory --> Hadoop DAPLAB avec SCP	Un fichier CSV du contenu de la table dans un dossier comportant le nom de la table	OK	L'outil mysql_to_csv.py doit être utilisé pour transformer en csv
106	Importation table avec l'outil bulk_import.py implémenté	La table Sqoop importée depuis MySQL est importée dans HDFS	OK	L'outil mysql_to_csv.py doit être utilisé pour transformer en csv
200	**Processing in Hadoop**			
201	Script 1 (6012_msds_stats.scala) en Scala sur Hadoop local	Enregistre le type de serveur (Samba, Win 2000, Win 2003 or 2008, Others) et le nombre dans un fichier	OK	
202	Script 1 (6012_msds_stats.scala) en Scala sur Hadoop DAPLAB	Enregistre le type de serveur (Samba, Win 2000, Win 2003 or 2008, Others) et le nombre dans un fichier	OK	
203	Script 2 (4013_build_smtp_vuln.scala) en Scala sur Hadoop local	Pour chaque date sur une année, enregistre dans un fichier le nombre de serveurs (Sendmail et Exim) à jour, pas à jour, inconnu	OK	
204	Script 2 (4013_build_smtp_vuln.scala) en Scala sur Hadoop DAPLAB	Pour chaque date sur une année, enregistre dans un fichier le nombre de serveurs (Sendmail et Exim) à jour, pas à jour, inconnu	OK	
205	Script 3 (0003_web_doctype.scala) en Scala sur Hadoop DAPLAB	Recense une statistique des DOCTYPE utilisés sur le Web suisse	OK	
206	Script 3 (0003_web_doctype.py) en Python sur Gateway stats	Recense une statistique des DOCTYPE utilisés sur le Web suisse	OK	
300	**Communication Gateway <=> Hadoop**			
301	Execution d'une commande ls par SSH sur Hadoop local (Connexion avec user/password)	Liste les fichiers du serveur Hadoop local	OK	
302	Execution d'une commande ls par SSH sur Hadoop DAPLAB (Connexion avec User/ Key_file/password_key)	Liste les fichiers du serveur Hadoop DAPLAB	OK	
303	Récupération d'un fichier par SCP depuis Hadoop local vers Gateway	Téléchargement du fichier	OK	
304	Récupération d'un fichier par SCP depuis Hadoop DAPLAB vers Gateway	Téléchargement du fichier	OK	
400	**Génération des graphiques**			
401	Génération d'un graphique en barres horizontal (6012_msds_stats)	Génère une image PNG du graphique en barre horizontale	OK	
402	Génération d'un graphique histogramme (4013_build_smtp_vuln)	Génère une image PNG du graphique histogramme 4013	OK	

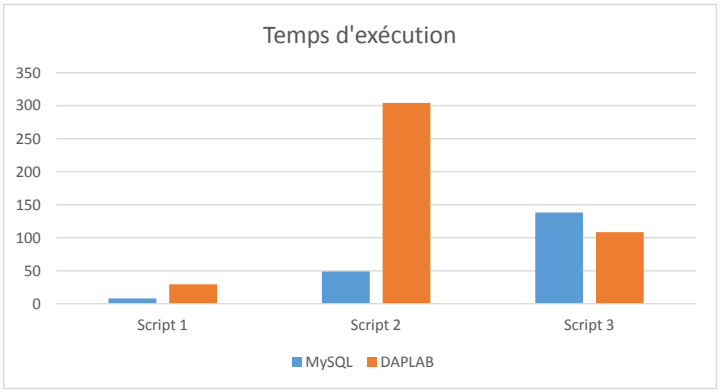
Script 1		Essai 1	Essai 2	Essai 3	Essai 4	Médiane
MySQL		47,009881	7,88050294	7,94123602	7,83069396	7,91086948
DAPLAB		28,576	29,86	33,688	28,942	29,401

Script 2		Essai 1	Essai 2	Essai 3	Essai 4	Médiane
MySQL		72,8034132	24,113997	23,76949	24,6019518	48,7159488
DAPLAB		307,631	303,849	304,283	303,042	304,066

Script 3		Essai 1	Essai 2	Essai 3	Essai 4	Médiane
MySQL		138,27374	16737,11	CRASH	CRASH	138,27374
DAPLAB		104,613	115,094	109,425	107,236	108,3305

Conditions de tests : 50 executeurs sur Spark-Shell

Tableau récapitulatif				
	Data[MB]	MySQL	DAPLAB	Remarques
Script 1	22,71	7,91086948	29,401	1 Fois petite quantité de données
Script 2	1219,8	48,7159488	304,066	24 Fois petites quantités de données
Script 3	3000	138,27374	108,3305	1 Fois grande quantité de données



MySQL

Script 1	Essai 1	Temps par étape
fetch	0,261780024	0,261780024
build	0,261869907	8,98838E-05
load	0,277369022	0,015499115
plot	0,277778864	0,000409842
End	8,276638985	7,998860121

Script 2	Essai 1	Essai 2	Essai 3	Essai 4	Temps par étape
fetch	30,04531193	0,728809834	0,772964001	0,745650053	0,745650053
build	454,079093	7,031751871	7,547644138	7,059988022	6,314337969
load	2759,631787	161,8009439	23,423177	11,53714299	4,47715497
plot	2765,326234	167,02648	28,75850415	16,75346017	5,216317177
End	2778,286451	173,612483	35,35304499	23,39816618	6,644706011

Script 3	Essai 1	Essai 2	Temps par étape
fetch	137,8518519	16736,54	137,8518519
build			
load			
plot	138,2737398	16737,11	0,421887875
End			