Evaluation de systèmes de masses de données dans le cadre d'analyse de logs

Pierre Rolland[1], Laurent d'Orazio[2][0000−0001−8614−1848], Frédéric Majorczyk[3], and Mohand Saïd Hacid[4]

[1] CentraleSupélec & IMT Atlantique, France
pierre.rolland@centraliens.net
[2] Univ Rennes, CNRS, IRISA, Lannion, France
laurent.dorazio@irisa.fr
[3] DGA-MI & CentraleSupélec, UMR 6074 (IRISA), France
frederic.majorczyk@intradef.gouv.fr
[4] Univ Lyon 1, CNRS, LIRIS, Villeurbanne, France
prenom.nom @univ-lyon1.fr

Résumé La supervision de sécurité des systèmes informatiques consiste à les analyser continuellement afin de détecter des évènements anormaux. Le développement des technologies engendre un déluge d'informations variées devant être traitées efficacement. Pour des raisons économiques et de limitations physiques les solutions traditionnelles de supervision de sécurité se limitent au stockage en silos et à des fenêtres temporelles récentes et réduites. L'informatique en nuage offre des solutions pour la gestion de masses de données. Cet article présente certaines de ces solutions et définit des problèmes liés à la gestion de masses d'historiques dédiées à la supervision de sécurité.

Keywords: Supervision de sécurité · Masses de Données

1 Introduction

La Lutte Informatique Défensive (LID) correspond à un champ de la cybersécurité dont l'objectif est de veiller à la sécurité d'un système ou d'un ensemble de systèmes en analysant des fichiers de journalisation (logs) afin de détecter des évènements suspects et/ou atypiques. La notion de système est dans ce contexte variable pouvant aller d'un système d'information à un véhicule militaire tel un avion de chasse ou un porte avion. Ainsi les informations, devant être traitées de manière Véloce, sont Volumineuses et Variées, faisant de la LID un cadre concret de données massives (Big Data) et des 3V.

Dans certains domaines d'application comme l'astronomie, la physique des particules ou encore les réseaux sociaux, les volumes sont si importants, qu'il est nécessaire de recourir à des environnements de traitement massivement parallèles (Massively Parallel Processing ou MPP). Ainsi se sont développés les systèmes de fichiers largement distribués (tels que GFS, HDFS), les environnements d'exécution massivement parallèles (MapReduce [2], Spark [1], etc.) au

dessus desquels ont été construits des systèmes d'analyse ou d'entreposage de données (Pig [5], Hive [9], etc.) pour faire face aux défis des données massives (Big Data).

Cet article présente nos travaux sur l'utilisation de solutions de masses de données pour la supervision de sécurité. Notre première contribution est une plateforme expérimentale reprenant l'environnement utilisé à l'Université de Rennes 1. Les expérimentations menées ont permis de finement configuré l'outil de manière à travailler sur le passage à l'échelle. La deuxième contribution est l'extension de cette plateforme avec le stockage sur système de fichier distribué HDFS et le moteur de traitement parallèle en mémoire Spark.

La suite de cet article est organisé de la manière suivante. La section 2 présente la plateforme expérimentale proposée, inspirée des solutions utilisées à l'Université de Rennes 1. La section 3 décrit l'architecture proposée pour la gestion à grand échelle. Enfin, la section 4 fournit des conclusions et perspectives.

2 Contexte

L'Université de Rennes 1 exploite l'outil Graylog [5] pour centraliser divers journaux de l'établissement. Il s'agit d'une solution opensource - de collecte et l'analyse de logs – reprenant les fonctionnalités d'autres SIEM tels que la suite Elastic ou Splunk. Par une interface web paramétrable, l'analyste accède à une vue centralisée des événements qui s'y déversent. Ces messages peuvent être filtrés, découpés par champs, redirigés ou valorisés par recoupement avec d'autres sources de données. L'analyste en sécurité est alors en mesure de détecter des anomalies, de paramétrer des alertes, d'opérer une surveillance quasi temps-réel depuis un «dashboard» ou de rechercher des événements à corréler (dans le cadre d'une analyse forensique par exemple). Graylog gère la partie authentification et l'épuration des logs au-delà d'un volume choisi ou d'un délai.

Un répartiteur de charge est placé en amont des instances Graylog. Les logs sont collectés depuis divers équipements et systèmes d'exploitation. Les formats acceptés en entrée sont donc variés : Syslog, Gelf, Raw (Plaintext). Le moteur d'indexation ElasticSearch [6] est paramétré pour conserver 3 mois d'historique à raison d'un indice par jour. Au delà, les données archivées hors d'ElasticSearch deviennent inaccessibles depuis Graylog.

Notre travail a permis une mise en pratique de l'outil Graylog v3.0.2 sur un environnement d'expérimentations. Les sections suivantes décrive le contexte expérimental (systèmes et logiciels) ainsi que les jeux de données utilisés.

2.1 Contexte expérimental

Notre plateforme d'évaluation (Graylog – ElasticSearch – Hadoop [7]) vise à reproduire l'architecture observée à l'Université de Rennes 1 avec les ressources

5. https ://www.graylog.org/
6. https ://www.elastic.co
7. https ://hadoop.apache.org

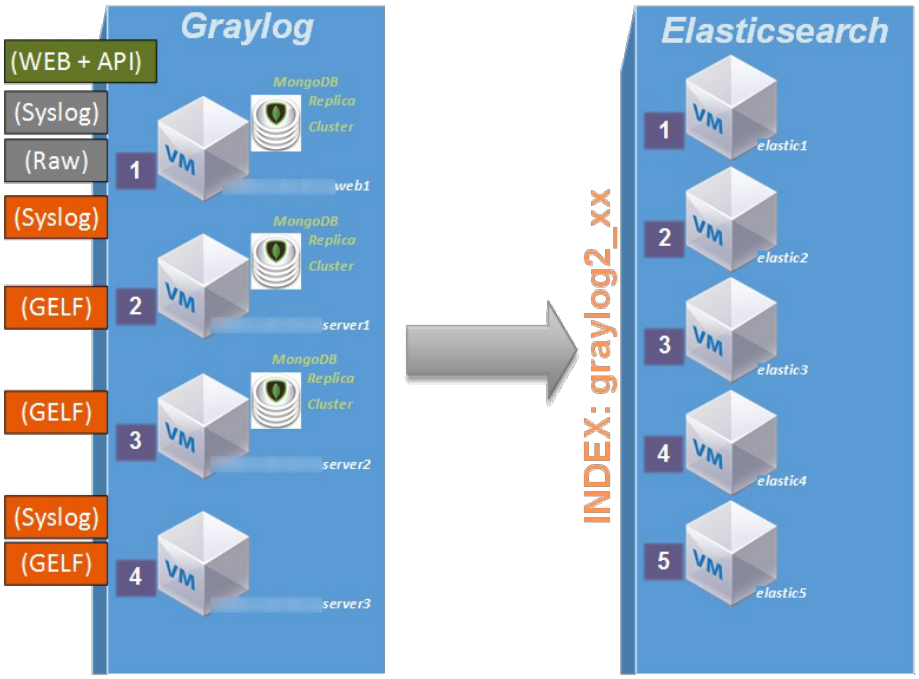

Figure 1. Architecture exploitée à l'Université de Rennes 1

matérielles suivantes : 6 hyperviseurs KVM/LXC (Bi-Xeon E5440@2.83GHz, 2 x 4 cores x 5652.54 bogomips), 16GB de mémoire ; lien réseau SFP+ (cartes Myricom Myri-10G) ; stockage sur disque local SSD ou montage d'images rados block device (Ceph RBD).

Les systèmes d'exploitation utilisés sont Linux Kernel 4.15 x86_64 (Debian 9 et Ubuntu 18.0.4 LTS). Les piles logicielles se composent de OpenJDK v1.8.0_212 pour assurer la compatibilité avec la version 6.8.1 d'ElasticSearch livrée avec Graylog 3.0.218 ; Ceph 12.2.12 Luminous ; HDFS, YARN et MapReduce en version 3.1.1 ; Tez v0.9.1 ; Hive v3.1.0 ; ZooKeeper v3.4.6 ; Spark2 v2.3.2 / Scala v2.11 (versions HDP packagées par HortonWorks19).

2.2 Jeux de données

Plusieurs jeux de données sont étudiés. Nous disposons d'un premier jeu de données d'entraînement constitués de logs iptables (232Mo) fournis en continuité de la thèse de David Pierrot). Le LANL propose également un jeu de données (10 Go compressé) composé d'événements variés [Kent2015].

Un autre jeu de données [12] du LANL propose «Network Event Data» au format CSV et des «Host Event Data» au format JSON. Notre analyse comparative porte sur ce dernier dataset. Une fois décompressée et regroupée, cette

collection forme un ensemble de 2,3 To. Compte tenu de la taille des données manipulées, nous comparons les empreintes des fichiers téléchargés à celles fournies par le LANL. Nous souhaitons à présent indexer ces deux types de données et réaliser des requêtes via Graylog. Compte-tenu des capacités de traitement et de stockage de notre serveur Graylog, nous n'intégrons que 11 jours sur les 90 proposés ; soit 914,355,585 événements. En contexte de production, ces 166,7 Go de données brutes doivent idéalement être répliqués par ElasticSearch et/ou par le système de fichiers sous-jacent.

2.3 Indexation

ElasticSearch (ES) est utilisé pour l'indexation de documents. Nos documents Graylog sont découpés en fragments (shards) répartis sur différents noeuds du cluster ES (es0, es1, ..., esN). Comme illustré par la figure 1, chaque noeud est un container LXC.

Le temps de réponse (en ms) est mesuré. Les mesures sont effectuées en recréant systématiquement un index ES pour les 166,7 Go (dataset du LANL). Ces données sont réparties vers un nombre croissant de noeuds sans être répliquées.

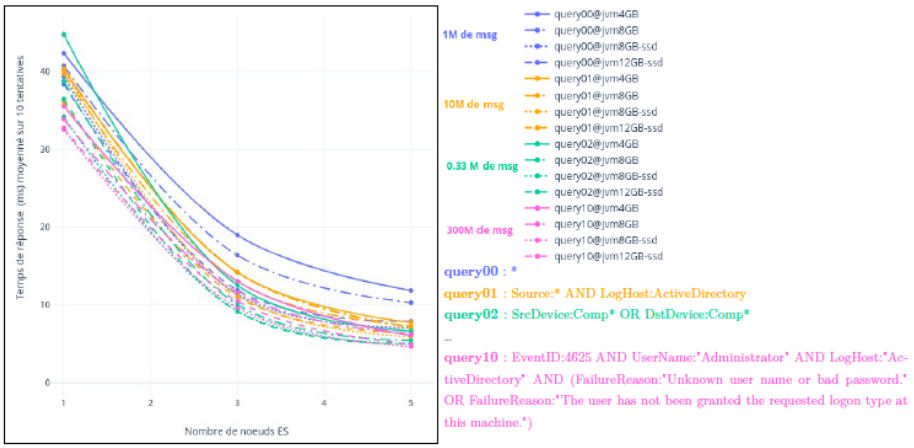

Figure 2. Temps de réponse en fonction du nombre de noeuds

L'absence de réplication nous permet de déterminer immédiatement l'avantage induit par l'ajout d'un noeud. Nous constatons que la plateforme permet un passage à l'échelle réalisé horizontalement : le déploiement de nouveaux noeuds renforce de manière très importante les performances du cluster ES comme en témoignent les mesures effectuées (figure 2).

Pour comparer visuellement les tendances, nous amplifions ou réduisons les résultats obtenus. À chaque résultat est appliqué un facteur de mise à l'échelle.

La requête query00 (qui retourne tous les événements) nous sert de référence d'échelle. L'indication "msg" correspond à la quantité de messages Graylog ("documents" selon la terminologie ES). À titre d'exemple, 5 noeuds ES (avec SSD et dont les JVM sont dotées de 12 Go de mémoire) sont capables de restituer 300 millions de messages en 4 ms pour une requête de type query10 correspondant aux échecs de connexion à l'AD en tant qu'administrateur.

L'utilisation d'un disque SSD local (par rapport à un volume Ceph RBD) et davantage de mémoire allouée à la JVM d'ElasticSearch (options -Xms,-Xmx) réduisent légèrement la latence quelles que soient les requêtes. Nous notons également une vitesse d'indexation accrue et une meilleure fluidité à la construction de nos Dashboards Graylog.

Le moteur Lucene [8] d'ES doit disposer d'une quantité de mémoire suffisante pour y placer toutes les valeurs de champs souhaitées. Nous allouons 50 % de la mémoire totale au Java heap. Au delà, les processus ES utilisent le swap (que nous désactivons). La mémoire non utilisée doit servir à la mise en cache des fichiers.

2.4 Motivations

Pour compenser son incapacité à réaliser des jointures entre indexes, ES propose des mécanismes de recherche avancée et imbriquée (nested). Ceci est fortement déconseillé compte-tenu du surcoût mémoire et calculatoire. Nous avons pu vérifier qu'un cluster ES correctement dimensionné permet de traiter davantage d'événements. Pour autant, plusieurs limites inhérentes à l'outil apparaissent.

Nous n'aborderons pas dans le cadre de ce travail les problèmes de cohérence de données ou de disponibilité. Nos travaux se focalisent sur le passage à l'échelle du stockage et du traitement des données. L'idée est ainsi d'étendre l'indexation avec ES, à l'aide de traitement massivement parallèle en mémoire avec Spark et du stockage sur système de fichiers distribués implémentant une répartition géographique des données, des stratégies de tiers de stockage chauds et froids (NVMe SSD, SAS He HDD).

3 Proposition

L'architecture proposée est illustrée par la figure 3. L'idée est de coupler les couches ELK (Elasticsearch, Logstash, Kibana) pourront avec des systèmes à grande échelle et en particulier l'éco-système Hadoop (HDFS et Spark).

3.1 Système de Fichiers Distribués

Apache Hadoop est un canevas logiciels pour le traitement de données massivement parallèle. Ce canevas comprend différents services tels que YARN (YetAnotherResourceManager) ou MapReduce [2].

8. https ://lucene.apache.org

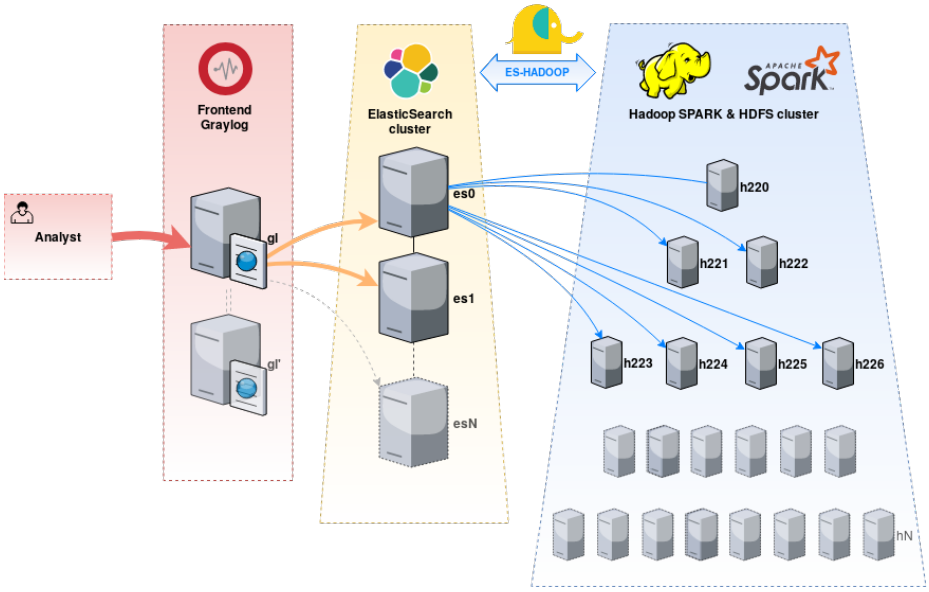

Figure 3. Architecture proposée

L'architecture proposée repose en particulier sur le service HDFS, un système de fichiers distribué dérivé de Google File System [4]. HDFS distingue deux types de noeuds : (1) le NameNode assure la coordination des noeuds de stockage et (2) les DataNodes, les noeuds de stockages. Les fichiers, en général de grande taille, sont ensuite découpés en morceaux, chaque morceau étant ensuite répartis (et souvent dupliqués) sur les différents DataNodes.

Dans le cadre de notre architecture, nous déployons plusieurs containers LXC (h208 à h227) dotés de 4 à 8 Go de mémoire et d'un seul coeur. Ces serveurs vont endosser le rôle de noeud de stockage HDFS ou de traitement (voir Section suivante).

3.2 Moteur de traitement de données

A l'instar de MapReduce ou encore Tez [7], Spark est un canevas de traitement de données pour des environnements parallèles à grande échelle dans des environnements sans partage [8]. Contrairement à MapReduce, la gestion des entrées/sorties des processus se fait en mémoire et non sur disque, permettant en général de réduire les temps d'exécution.

Pour ce faire, Spark repose sur le concept de RDD (Resilient Distributed Datasets) [13], structures de données distribuées, gérées en mémoire et immutable. Des schémas peuvent être appliqués au dessus des RDD, de manière à créer des DataFrames pouvant être interrogés à l'aide de requêtes déclaratives en SQL via

Spark SQL. Au sein de notre architecture, les fichiers CSV ou JSON du LANL sont ainsi convertis en DataFrames à partir de HDFS.

Les RDD décrits ci-dessus peuvent être issus d'un accès HDFS, mais ils peuvent également être créés depuis ElasticSearch. Le connecteur ES-Hadoop et sa librairie sparkelasticsearch disposent de méthodes esRDD et saveToEs. Celles-ci permettent de réaliser des lectures/écritures sur les indexes ES de Graylog. Ce plugin apporte une capacité de traitement temps réel que ne permet pas le gestionnaire de batches Hadoop MapReduce.

Le coordinateur, ou maître, de la couche Spark déployé est le conteneur h205. Comme évoqué précédemment, les noeuds responsables du traitement de données sont correspondent à des conteneurs LXC (h208 à h227) dotés de 4 à 8 Go de mémoire et d'un seul coeur.

4 Conclusion

Cet article a présenté l'apport des systèmes de gestion de masses de données pour l'analyse et le traitement massifs d'événements de sécurité. Il considère les pratiques de l'Université de Rennes 1 et de l'expérience d'acteurs de la cybersécurité. Une plateforme d'évaluation a pu être déployée avec une interface de consultation-requête (Graylog) et un moteur d'indexation (ElasticSearch) traduisant une infrastructure de gestion de logs NoSQL traditionnelle. Nous y avons intégré des jeux de données du domaine afin de les manipuler. Le recours à ces outils de stockage distribué et de traitement de masses de données apporte une solution concrète au problème soulevé, à savoir, permettre la fouille de données massives parmi des événements de sécurité dans une perspective d'aide à la décision, notamment en contexte de lutte informatique défensive.

Les perspectives de travaux complémentaires sont nombreuses. Des travaux sont ainsi actuellement en cours sur les jointures floues à grande échelle [11]. Il s'agit d'optimiser le traitement de ces opérations en utilisant notamment des structures approximatives tels que les filtres de Bloom. Une autre direction de recherche correspond au traitement de requêtes sur données chiffrées [10]. Une étude est également menée sur l'accélération matérielle à l'aide de FPGA [3]. Enfin, un autre axe concerne la réalisation de bancs d'essais pour systèmes de supervision (à grande échelle). Il n'existe en effet que très peu de bancs d'essais pour les systèmes de gestion de masses de données (et certaines spécificités, notamment l'imbrication autorisée [6]) et encore moins pour le cas précis de la Lutte Informatique Défensive.

Remerciements

Merci à David Pierrot, Naïg Ledaim–Olivier et Enora Morvan pour les échanges autour de ce projet.

Références

1. Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., Zaharia, M. : Spark {SQL :} Relational Data Processing in Spark. In : Proceedings of the SIGMOD International Conference on Management of Data. pp. 1383–1394. Melbourne, Victoria, Australia (2015)
2. Dean, J., Ghemawat, S. : MapReduce : simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008)
3. d'Orazio, L., Lallet, J. : Semantic caching framework, an application to fpga-based application for iot security monitoring. Open Journal of Internet of Things (OJIOT) **4**(1), 150–157 (2018)
4. Ghemawat, S., Gobioff, H., Leung, S.T. : The Google file system. In : Proceedings of the Symposium on Operating Systems Principles (SOSP). pp. 29–43. Bolton Landing, NY, USA (2003)
5. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A. : Pig latin : a not-so-foreign language for data processing. In : Proceedings of the SIGMOD International Conference on Management of Data. pp. 1099–1110. Vancouver, BC, Canada (2008)
6. Pilven, M., Scherzinger, S., d'Orazio, L. : On complex value relations in hive. In : Proceedings of the International Workshop on Modeling and Management of Big Data (MOBID@ER). Salvador, Bahia, Brazil (2019)
7. Saha, B., Shah, H., Seth, S., Vijayaraghavan, G., Murthy, A.C., Curino, C. : Apache Tez : A Unifying Framework for Modeling and Building Data Processing Applications. In : Proceedings of the SIGMOD International Conference on Management of Data. pp. 1357–1369. Melbourne, Victoria, Australia (2015)
8. Stonebraker, M. : The Case for Shared Nothing. IEEE Database Engineering Bulletin **9**(1), 4–9 (1986)
9. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R. : Hive - {A} Warehousing Solution Over a Map-Reduce Framework. Proceedings of the Very Large Data Bases Endowment (PVLDB) **2**(2), 1626–1629 (2009)
10. Tran, H.V., Allard, T., d'Orazio, L., El Abbadi, A. : Range query processing for monitoring applications over untrustworthy clouds. In : Proceedings of the International Conference on Extending Database (EDBT). Lisbon, Portugal (2019)
11. Tran, T., Phan, T., Laurent, A., d'Orazio, L. : Improving hamming distance-based fuzzy join in mapreduce using bloom filters. In : Proceedings of the International Conference on Fuzzy Systems, (FUZZ-IEEE). pp. 1–7. Rio de Janeiro, Brazil (2018)
12. Turcotte, M.J.M., Kent, A.D., Hash, C. : Unified host and network data set. In : Data Science for Cyber-Security, pp. 1–22 (2018)
13. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M.J., Shenker, S., Stoica, I. : Resilient Distributed Datasets : A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In : Proceedings of the {USENIX} Symposium on Networked Systems Design and Implementation (NSDI). pp. 15–28. San Jose, CA, USA (2012)