B.U.T. Informatique

Parcours A
Année 2023

Rapport de stage :
Étude de la consommation énergétique d'un Data Lake

Étudiant : Alexandre Maurice

Stage réalisé à **Domolandes**

Date :
Du 17 avril 2023 au 23 juin 2023

Encadrement :
M. Hernán Humberto Alvarez Valera

Lien vers la publication (HAL)

Enseignant référent : M^me^ Margaret Borthwick

Lecteur : M^me^ Nathalie Valles-Parlangeau

Année universitaire 2022-2023 (Semestre 4)

Abstract. This internship took place at Domolandes, a technopole focused on sustainable construction. The internship involved working in the unique environment of research, with a research topic exploring the energy consumption of Data Lakes. Data Lakes are raw data structures that can have highly heterogeneous formats, designed to facilitate work on Big Data Analytics, including Business Intelligence, Machine Learning, and more. The main objective was to establish a proof of concept environment for a Data Lake in order to make energy consumption measurements. The laboratory decided to select a Data Lake with a benchmark from scientific literature, that would serve as the basis for the study. Automating the deployment process was a crucial aspect, emphasizing the importance of system configuration and scripting. Simultaneously, measuring energy consumption played a vital role in assessing the ecological impact of Data Lakes. Beyond monitoring the CPU (which was the main component targeted by previous studies), measurements included RAM, network interfaces, and storage devices, providing a wider comprehensive evaluation of the environmental footprint. These measurements served two primary purposes. First, they enabled real-time monitoring of energy consumption, facilitating in-depth analysis of consumption trends and their sustainability implications. Second, the results are meant for the groundwork of a forthcoming publication on the concept of an "Eco-friendly Data Lake", emphasizing the need to consider energy consumption of all relevant system components for environmentally responsible Data Lake architectures.

In summary, this internship immersed me in the research environment, allowing me to develop crucial skills in systems and data management, and to discover the world of scientific research.

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements à M. Philippe Roose qui a proposé cette offre de stage aux étudiants et m'a permis de postuler dans cette entreprise. Cela m'a permis d'intégrer un stage dans le domaine de la recherche, domaine que je ciblais tout particulièrement.

Je tiens aussi à remercier vivement mon maître de stage, M. Humberto Valera pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce aussi à sa confiance, j'ai pu accomplir au mieux mes missions. Il a su rester tout le temps à l'écoute et être force d'aide et de conseils.

Je suis aussi reconnaissant envers toute l'équipe de recherche de l'Université de Pau et des Pays de l'Adour ainsi que de l'université Toulouse Capitole, pour leur soutien et leur contribution tout au long de mon stage. Leurs connaissances et leurs discussions ont été inestimables dans le développement de mes compétences et de ma compréhension du domaine de la recherche.

Par ailleurs, je tiens à exprimer ma gratitude envers l'équipe de Domolandes pour leur accueil chaleureux, leur attention ainsi que leur précieuse assistance tout au long de mon stage.

Table des matières

1 Introduction **6**
1.1 Motivation personnelle 6
1.2 Présentation de l'entreprise 7
1.3 Organisation de l'entreprise 8
1.4 Environnement du stage 8
1.5 Mission du stagiaire 9
1.5.1 Intitulé de mission original 9
1.5.2 Reformulation des missions par l'étudiant 9

2 Méthodologie **10**
2.1 L'origine des recherches 10
2.2 La finalité 10
2.3 La provenance de la consommation énergétique 10
2.4 Les technologies utilisées 11
2.4.1 Introduction au concept de Data Lake 11
2.4.2 La différence avec les SGBD traditionnels et les Data Warehouses 12
2.4.3 Les opérations réalisées par un Data Lake 12
2.4.4 AUDAL, l'architecture de Data Lake choisie 13

3 Production **14**
3.1 La mise en place du Data Lake AUDAL 14
3.1.1 Étude d'AUDAL 14
3.1.2 Solutions proposées 15
3.1.3 Difficultés rencontrées 16
3.1.4 Preuves 17
3.2 Développement d'une solution de monitoring en temps réel . . . 18
3.2.1 Conception 18
3.2.1.1 Choix des technologies 19
3.2.2 Difficultés rencontrées 20
3.2.3 Visualisation finale 21
3.3 Scripts de mesure énergétique 23
3.3.1 Déploiement de PowerJoular 23
3.3.2 Déploiement de Scaphandre 23
3.3.3 Validation des scripts de mon tuteur 24
3.3.4 Conclusion 29

4 Bilans **30**
4.1 Bilan général 30
4.2 Compétences du B.U.T. 30
4.3 Perspectives professionnelles 32

Références **33**

Table des figures

1 Situation géographique de Domolandes 7
2 Organigramme de Domolandes au 7 juin 2023 8
3 Architecture générale d'un Data Lake selon Orogat (2021) 12
4 L'architecture de AUDAL 13
5 L'organisation des métadonnées intra-objet de AUDAL 15
6 docker-compose pour AUDAL 16
7 Architecture d'EnergyMonitor 19
8 Surveillance de l'entièreté de la machine avec EnergyMonitor . . 21
9 Surveillance du processus **systemd** avec EnergyMonitor 22
10 Extrait du résultat de la commande **pstree** sous Linux 26
11 Schématisation de la puissance utilisée par Firefox (à l'instant T0) puis par ses sous processus (à un instant Tx) 27
12 Schématisation de la somme successive de la puissance utilisée par Firefox et de ses processus sur un intervalle de temps 27
13 Schématisation du temps CPU alloué à chaque processus et de la tentative de lecture des informations de consommation en parallèle 28

Liste des tableaux

1 Liste des membres de chaque partie 8
2 Avantages et inconvénients de chaque outil de mesure 29

1 Introduction

1.1 Motivation personnelle

Avant même de commencer mes recherches de stage, je m'orientais déjà pour un stage qui serait extrêmement lié au domaine de la recherche. C'est un domaine qui m'attire et dans lequel j'envisage de poursuivre mes études.

Bien que j'aie des compétences qui sont, selon moi, plus orientées vers les méthodes algorithmiques plutôt que dans une connaissance relativement vaste du domaine des systèmes, cela ne m'a pas rebuté pour postuler chez Domolandes. Au contraire, j'ai vu cela comme une opportunité d'en apprendre plus sur ce domaine et d'augmenter significativement mon agilité dans cet environnement, ce qui augmenterait très largement l'ensemble de mes compétences.

J'étais aussi intéressé par le domaine du Big Data, et savais pertinemment qu'un stage dans le domaine allait me faire découvrir plus en profondeur de nouvelles structures de données, qui est toujours quelque chose que je ne peux qu'apprécier.

En tant que préambule, j'aimerais ajouter que ce rapport de stage rentre dans des détails que je juge comme relativement techniques et sur lesquels il n'est pas nécessaire pour vous, lecteur, de vous y attarder. Cependant, l'évocation de telles précisions est de ma volonté, puisqu'il est pour moi primordial pour se rapprocher au plus possible du niveau de complexité étudié dans le domaine de nos recherches, afin de représenter correctement mon travail. De plus, j'apprécie énormément la complexité de ces systèmes articulés et il serait selon moi regrettable de ne pas décrire ces concepts fondamentaux. J'ai aussi tenu à inclure ces précisions pour rendre compte de mes progrès sur ces connaissances techniques dans les domaines évoqués, en qu'il est souvent rendu difficile de rendre compte de ce genre de progrès par des preuves tangibles comme des captures d'écran ou des résultats qui ne sont pas visiblement austères.

1.2 Présentation de l'entreprise

Domolandes est un pôle technologique créé en septembre 2011 dédié à la construction durable. Fruit d'un partenariat entre le Conseil général des Landes et la Communauté de Communes de Maremne Adour Côte Sud, il se situe dans le parc d'activités Atlantisud à Saint-Geours-de-Maremne (40230). Fondé sur les principes du **développement durable**, Domolandes vise à promouvoir des projets améliorant la qualité de vie tout en minimisant leur impact sur l'environnement rural ou urbain et en gérant de manière responsable les ressources.

Domolandes dispose d'une pépinière d'entreprises innovantes, offrant un environnement propice au développement des jeunes entreprises. Elles bénéficient d'un soutien personnalisé, de services partagés et d'un accès à un réseau d'experts et de partenaires, favorisant leur croissance et leur succès.

En plus de la pépinière, Domolandes abrite un hôtel d'entreprises fournissant des locaux fonctionnels et professionnels pour les activités des entreprises présentes sur le site. Les locataires bénéficient d'infrastructures de qualité, de services communs et de synergies avec les autres entreprises présentes.

Acteur clé de la construction durable, Domolandes favorise l'innovation, la collaboration et le développement de solutions écologiques. En rassemblant entreprises, experts, chercheurs et acteurs du secteur, il contribue à l'émergence de projets et de technologies innovantes répondant aux enjeux de la transition écologique et énergétique.

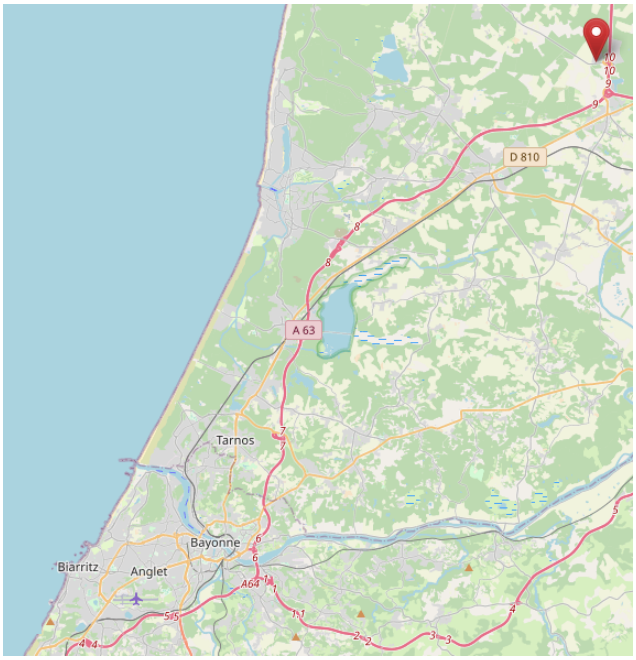

FIGURE 1 – Situation géographique de Domolandes

1.3 Organisation de l'entreprise

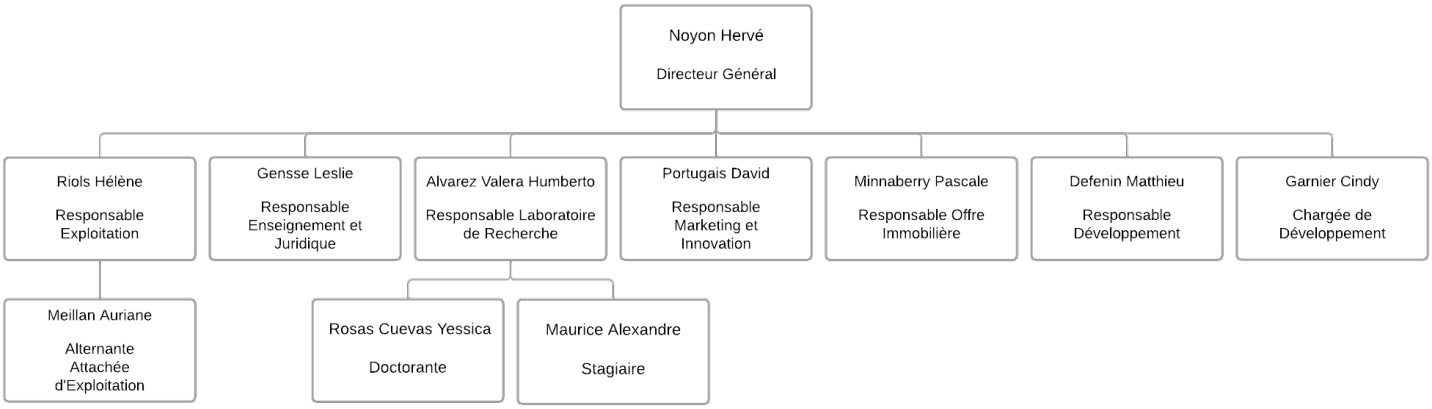

FIGURE 2 – Organigramme de Domolandes au 7 juin 2023

1.4 Environnement du stage

Mon stage est réalisé dans le pôle du Laboratoire de recherches de Domolandes, dans le cadre d'un partenariat entre l'Université de Pau et des Pays de l'Adour (UPPA) et l'Université Toulouse 1 Capitole (UT1). De ce fait, l'environnement du stage se passe dans un service de R&D, qui sort du cadre de celui de l'entreprise pour entrer dans un domaine beaucoup plus exploratoire, et nécessitant de collaborer étroitement avec le reste de l'équipe de recherche pour comprendre la perspective globale des différents projets.

Entité	Membre	Rôle
Domolandes	M. Hernán Humberto Alvarez Valera	Responsable de la Recherche et du Développement
	M^{me} Yessica Rosas Cuevas	Doctorante
	M. Alexandre Maurice	Stagiaire
UPPA	M. Philippe Roose	Enseignant chercheur
	M^{me} Nathalie Valles-Parlangeau	Enseignant chercheur
UT1	M. Jiefu Song	Enseignant chercheur
	M. Franck Ravat	Enseignant chercheur

TABLE 1 – Liste des membres de chaque partie

1.5 Mission du stagiaire

1.5.1 Intitulé de mission original

Sujet initial :

Dans le cadre d'une collaboration de recherche entre le technopole Domolandes à Saint Geours de Maremne et le LIUPPA (Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour), le stagiaire travaillera dans le domaine des lacs de données (plateforme de gestion de données massives distribuées et hétérogènes).

Missions :

— à l'aide de l'application PowerJoular, réaliser des scripts de mesures énergétique des processus d'ingestions de données dans l'architecture de Data Lakes à mettre en œuvre
— réaliser des scripts de déploiement de Data Lakes
— implémenter du reporting de données sur des tableaux de bord type Grafana (ou autre)

1.5.2 Reformulation des missions par l'étudiant

Il s'est révélé que nous n'allions pas utiliser uniquement PowerJoular pour les mesures de consommation. Je généralise donc à "utiliser des outils de mesures énergétique de processus". Plus de détails seront livrés dans la section 3.3.

Sujet :

Le fil directeur du projet est de pouvoir publier une étude concernant la consommation énergétique d'un Data Lake, caractérisé par des "opérations type" décrites dans la section 2.4.3. Il m'est aussi demandé de proposer une solution pour permettre une visualisation en temps réel de l'évolution de la consommation par processus.

Missions :

— mettre en place un Data Lake et dérouler ses multiples opérations type
— à l'aide d'outils de mesure énergétique, réaliser des scripts de mesures énergétique des opérations type réalisées l'architecture de Data Lakes mise en place
— automatiser le déploiement du Data Lake et relever en parallèle le coût énergétique des opérations
— proposer une visualisation en temps réel ainsi que divers tableaux de bord grâce à Grafana, un outil d'analyse et de présentation de données spécialisé

2 Méthodologie

2.1 L'origine des recherches

L'objectif était initialement, aux alentours de l'année 2020, de créer une chaire industrielle entre l'UT1 et Domolandes, dans le cadre de l'analyse des données massives. Après une présentation de M. Philippe Roose à Domolandes, où était introduit le *Green IT*, l'entreprise a décidé de greffer cette thématique au projet, et de l'illustrer avec le nom de "bien vivre et bien vieillir dans le territoire", aux alentours d'octobre 2021. Tout ce processus a entraîné une série de travaux autour de la thématique du *Green IT*, et m'a permis d'obtenir cette opportunité de stage chez Domolandes.

2.2 La finalité

L'objectif des études est, à terme, de pouvoir proposer un Data Lake éco-responsable, à la différence de ceux généralement présents actuellement. Une publication du genre dans ce domaine est une grande avancée et constituerait en une première pierre dans la transition écologique au niveau des Data Lake.

La méthode d'approche proposée par mon tuteur et décrite dans Alvarez Valera (2022) consiste à répartir la charge de travail entre chaque machine physique d'un cluster (ensemble de machines communiquant à travers le réseau) en choisissant la tâche que va réaliser chaque machine du cluster. Cette méthode de *load balancing* (ou "répartition de la charge") nécessite de récupérer en temps réel les données de consommation de chaque machine afin de pouvoir choisir la machine la plus apte à réaliser l'opération demandée.

2.3 La provenance de la consommation énergétique

Lorsque nous parlons de la consommation énergétique d'un ordinateur, nous devons comprendre que la source de cette consommation provient principalement des composants matériels qui permettent à l'ordinateur d'exécuter les tâches que nous lui demandons. Ces composants sont complexes et ont chacun leur rôle à jouer dans le fonctionnement, et donc, dans la consommation d'un appareil.

Chaque tâche effectuée sur un ordinateur peut être identifiée comme une entité nommée "processus", et chaque processus, aussi minime soit-il, consomme des ressources sur la machine. Les tâches demandées ont un impact sur le niveau de stress de chaque composant matériel, comme par exemple du rendu vidéo ou des simulations qui ont tendance à solliciter de manière intensive la carte graphique, entraînant alors une consommation d'énergie plus élevée.

Dans un Data Lake Dans notre domaine d'étude, les opérations uniques sont généralement extrêmement lourdes, et sont répétées sur une très longue durée. Les opérations décrites dans 2.4.3 transfèrent des volumes de données massifs sur des types complexes, réalisant alors tout un pipeline de calcul articulé stressant énormément les composants. Tout cela questionne sur les implications directes de ces opérations, et pousse à réfléchir sur les possibilités d'améliorations des stratégies de gestion de ces clusters, ainsi que de l'aversion à la perte de ces dernières. L'objectif vise à proposer une solution utilitariste pour améliorer la situation actuelle d'un point de vue de la consommation énergétique.

Plus de détails sur le fonctionnement des processus seront livrés dans le chapitre 3.3.3.

2.4 Les technologies utilisées

2.4.1 Introduction au concept de Data Lake

Un lac de données, également appelé Data Lake en anglais, est une méthode de stockage utilisée dans le domaine du Big Data, qui désigne la gestion de grandes quantités de données. Contrairement à d'autres méthodes de stockage, un lac de données conserve les données dans leur **format d'origine** (ou *RAW data*). L'objectif principal du lac de données est de permettre un stockage rapide et volumineux de données provenant de différentes sources, en adoptant une architecture en cluster.

Dans un lac de données, vous pouvez trouver différents types de données provenant de sources variées [3] :

- **Des données structurées** : généralement issues de bases de données relationnelles et organisées sous forme de lignes et de colonnes.
- **Des données variées** : provenant de bases de données NoSQL, qui peuvent prendre différentes formes et structures selon les besoins spécifiques de chaque application.
- **Des données semi-structurées** : les fichiers CSV, les journaux, les fichiers XML ou JSON, qui ne suivent pas une structure rigide mais contiennent des informations organisées de manière plus flexible.
- **Des données non structurées** : les fichiers PDF, qui ne suivent aucune structure prédéfinie et nécessitent des méthodes spécifiques pour les analyser.
- **Les fichiers de type BLOB (Binary Large OBject)** : images ou fichiers audio ou vidéo, qui sont stockés en tant que objets binaires.

En résumé, un lac de données est une méthode de stockage qui permet de gérer des quantités massives de données provenant de sources différentes, tout en préservant les formats d'origine. Cela offre une grande flexibilité et ouvre des possibilités d'analyse approfondie des données pour les entreprises et les chercheurs qui s'intéressent au domaine du Big Data.

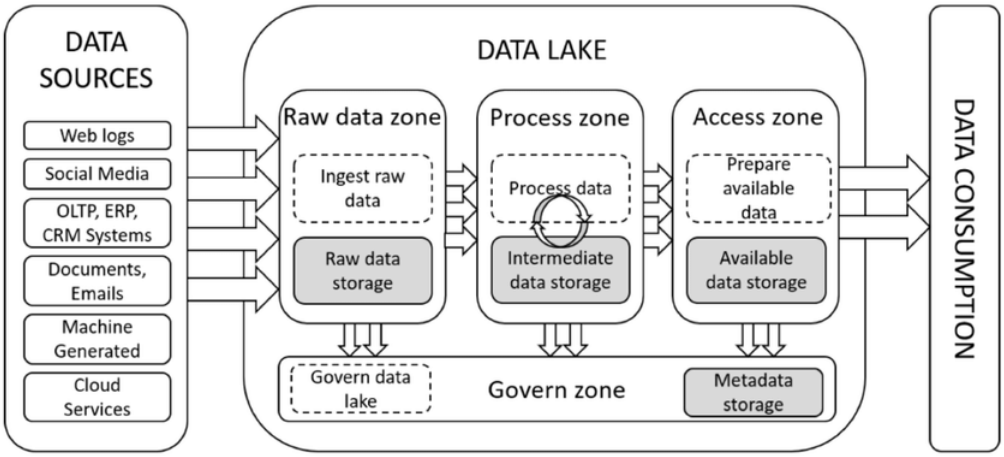

FIGURE 3 – Architecture générale d'un Data Lake selon Orogat (2021)

2.4.2 La différence avec les SGBD traditionnels et les Data Warehouses

Les SGBD traditionnels Les Systèmes de Gestion de Bases de données (SGBD) traditionnels travaillent sur des structures avec des schémas prédéfinis, optimisées pour des requêtes spécifiques et utilisées principalement pour des applications transactionnelles. En revanche, un Data Lake stocke des données brutes sans schéma prédéfini, offrant flexibilité et extensibilité pour intégrer des données provenant de différentes sources.

Les entrepôts de données Les entrepôts de données (ou *Data Warehouse en anglais*), quant à eux, transforment et organisent des données dans un schéma prédéfini pour l'analyse et les rapports. Les Data Lakes se distinguent des entrepôts de données par leur nature non structurée, permettant de stocker les données dans leur format brut sans transformation immédiate.

2.4.3 Les opérations réalisées par un Data Lake

Ces opérations sont variées et complexes. Elles peuvent être constituées d'un seul processus principal tout comme d'une série de processus très courts pour chaque document par exemple.

Voici une liste non-exhaustive des opérations effectuées :

— **Ingestion** : Collecte de données brutes de différentes sources.
— **Stockage** : Stockage évolutif des données brutes, sans schéma prédéfini.
— **Catalogage et métadonnées** : Organisation des données avec des informations essentielles.
— **Transformation** : Nettoyer et transformer les données selon les besoins.
— **Analyse** : Exploration approfondie des données, requêtage...

2.4.4 AUDAL, l'architecture de Data Lake choisie

AUDAL veut dire "AURA-PMI Data Lake", un Data Lake réalisé pour la région Auvergne-Rhône-Alpes (AURA), spécialement pour les Petites et Moyennes Industries (PMI). Son architecture est décrite dans dans Sawadogo et al. (2021) [4]. Nous avons choisi de l'utiliser puisqu'elle était était relativement simple à mettre en place et qu'elle convenait aux besoins de l'étude. En plus de cela, l'équipe l'ayant développé a aussi proposé une solution permettant de télécharger des documents textuels au format PDF et des tableaux au format CSV, destinés à être ingérés dans le Data Lake dans le futur. Enfin, l'étude de Sawadogo and Darmont (2021) propose aussi un banc d'essai ("benchmark" en anglais) avec des requêtes type à effectuer une fois que la structure est complète afin de la mettre à l'épreuve.

Il est intéressant de noter que ces études ont été présentées comme le *proof of concept* d'une structure de Data Lake, et que les essais étaient orientés vers le temps d'exécution de certaines requêtes plutôt que d'évaluer la consommation énergétique des opérations.

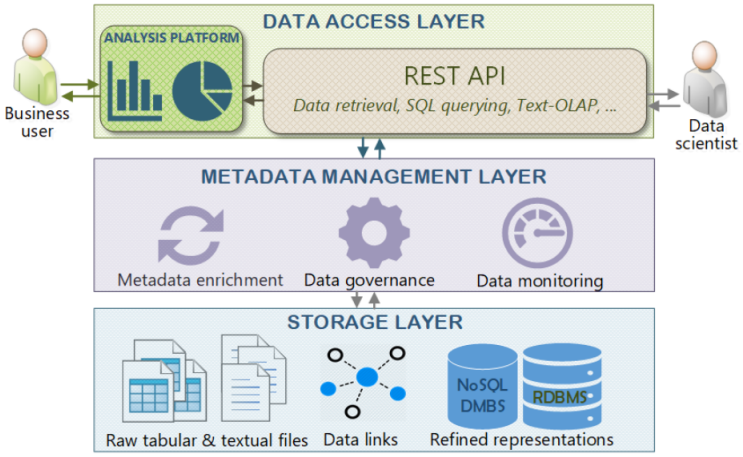

FIGURE 4 – L'architecture de AUDAL

3 Production

Cette partie sera organisée selon les deux principales missions qui m'ont été données, mais aussi d'une troisième mission annexe très importante pour la validité des expériences. Ces trois missions ont pu se réaliser en parallèle pendant le stage.

3.1 La mise en place du Data Lake AUDAL

3.1.1 Étude d'AUDAL

Les technologies utilisées dans AUDAL sont des services qui sont activés en tâche de fond sur une machine ou bien sur des machines distantes dans le cadre d'un cluster. Dans notre cas, nous travaillons en local, c'est-à-dire que tout est déployé sur une seule et même machine.

Les services utilisés sont les suivants :

- **Elasticsearch** : un logiciel utilisé pour l'indexation et la recherche de données. Il fournit un moteur de recherche distribué et multientité à travers une API REST (interface qui permet aux systèmes informatiques de communiquer entre eux en utilisant le protocole HTTP).
- **Neo4j** : un SGBD orienté graphe, qui permet de représenter les données en tant que nœuds reliés par un ensemble d'arcs. Chacun de ces objet posséde ses propres propriétés.
- **MongoDB** : un SGBD orienté graphe, ne nécessitant pas de schéma prédéfini des données.
- **SQLite** : un moteur de base de données relationnelle accessible par le langage SQL. Il diffère des serveurs de bases de données traditionnels (comme MySQL ou MariaDB par exemple) en qu'il est intégré directement aux programmes et stocke l'intégralité de la base de données dans un fichier en local.

Il est intéressant de noter qu'**Elasticsearch** n'est pas présent de manière explicite dans l'organisation des métadonnées intra-objet de AUDAL [5] puisqu'il crée des liens entre les documents, de manière "affinée" ("refined" dans la figure) et ne constitue donc pas un service en soi.

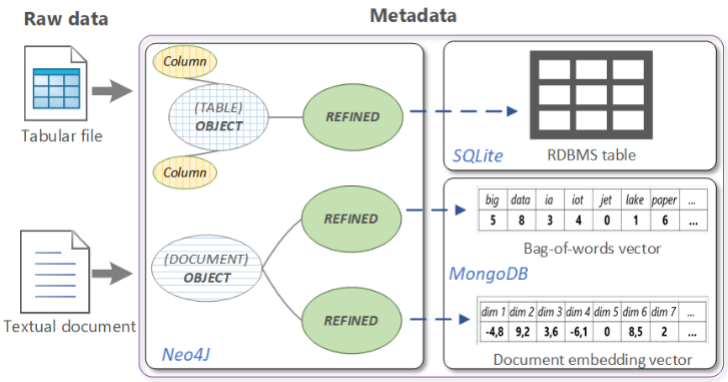

FIGURE 5 – L'organisation des métadonnées intra-objet de AUDAL

3.1.2 Solutions proposées

Au départ, les services étaient déployés tels quels sur la machine que nous utilisions pour réaliser les expériences, mais cela s'est trouvé être sous-optimisé pour le développement incrémental du projet, en raison des multiples lancements du projet qui nécessitaient de nettoyer à chaque fois les bases de données, notamment de **Neo4j**, ce qui prenait un temps conséquent. Pour plus de portabilité, j'ai proposé de conteneuriser ces services à l'aide de Docker et ai donc réalisé un docker-compose simple [6]. On notera que SQLite n'est pas présent dans ce fichier, simplement car c'est un moteur qui n'a besoin que d'un fichier de base de données pour fonctionner, comme mentionné dans 3.1.1.

```
version: '3'

services:
  neo4j:
    image: neo4j:4.3.2
    volumes:
      - ./neo4j_data:/data
    ports:
      - 7474:7474
      - 7687:7687

  mongo:
    image: mongo:4.4.6
    volumes:
      - ./mongo_data:/data/db
    ports:
      - 27017:27017

volumes:
  neo4j_data:
  mongo_data:
```

FIGURE 6 – docker-compose pour AUDAL

3.1.3 Difficultés rencontrées

Conception La première étape aurait dû se concentrer sur la compréhension de l'architecture d'AUDAL. Cela n'a pas été réalisé dans un premier temps car nous voulions avoir des résultats le plus rapidement possible pour une publication avant mi-mai. Cet objectif a échoué dû à des problèmes de compréhension de ladite architecture, et d'autres problèmes mentionnés dans 3.

Programmation AUDAL est une solution qui a été proposée en 2021, et dans un souci de facilité, les programmes développés utilisaient des fonctions et types dépréciés. Les programmes d'ingestion de données ont nécessité énormément d'adaptation. De plus, les technologies utilisées sont des technologies qui bénéficient d'une communauté extrêmement active, et le nombre de nouvelles versions sorties entre temps est très conséquent, entraînant alors des problèmes de sécurité, de rétro-compatibilité des services et de fonctionnement général.

La phase de débogage a pris, mis bout à bout, deux semaines environ, mais a en réalité duré sur toute l'étendue du stage, précisément à cause des problèmes susmentionnés.

Si **Elasticsearch** n'est pas présent dans le docker-compose [6], c'est aussi en raison des droits d'accès. Elasticsearch est une interface qui est censée pouvoir ingérer des données, renvoyer le résultat de requêtes, mais n'autorise pas la suppression du fichier de la base de données. Pour ne pas forcer la suppression avec

un processus externe, complexifiant la structure et assurant moins de fiabilité sur le long terme, j'ai préféré laisser Elasticsearch en tant que service sur la machine, en local, sans le conteneuriser.

3.1.4 Preuves

Il est compliqué de fournir des captures d'écran puisque le travail réalisé dans cette partie n'est pas graphique. Dans un souci de détail, les diagrammes d'architecture suffisant à décrire l'environnement mis en place mais restants toutefois très austères, je vais détailler le processus et les technologies utilisées ici :

- **Obtention des données brutes** : DLBench (Sawadogo and Darmont (2021)) est constitué de deux programmes **Python** permettant de télécharger des fichiers "type" pour être ensuite manipulés dans AUDAL. Le premier script sert à télécharger des documents textuels PDF issus de l'archive HAL (`https://hal.science/`), tandis que le second permet d'extraire des données tabulaires CSV d'une base de données SQLite.
- **Ingestion** : Le dépôt Github Sawadogo (2021b) contient deux parties. La première permet d'ingérer les données dans le Data Lake, sans les transformer. Elle est programmée en Java sous la forme d'un projet Netbeans (un environnement de développement intégré ou IDE). La deuxième partie est elle programmée en Python (plus précisément dans un Jupyter-Notebook) et permet de réaliser une série de traitements sur ces données, comme de la lemmatization ou classement sémantiques.
- **Analyse** : Finalement, la mise en place d'AUDAL (Sawadogo (2021a)) est réalisée à travers l'API-AUDAL, et permet de réaliser des requêtes qui peuvent être adressées à travers un outil en interface graphique comme Postman. Malheureusement, à l'heure de la rédaction de ce rapport, nous ne sommes pas parvenus à déployer cette étape.

3.2 Développement d'une solution de monitoring en temps réel

3.2.1 Conception

J'ai tout d'abord commencé à réaliser les spécifications externes de l'application qu'il m'était demandé de développer. L'idée est de choisir le meilleur compromis parmi toutes les spécifications, en évitant de choisir une technologie qui entraverait un autre critère.

Les spécifications étaient les suivantes :

— **Portabilité** : un système déployable facilement sur n'importe quelle machine
— **Performances** : un système peu gourmand en ressources puisque les traitements réalisés en parallèle sont relativement lourds, et qu'il faut éviter de créer de la concurrence inutile entre les processus, particulièrement dans ces cadres de *benchmarks*.
— **Stockage** : une *Time Series Database* (TSDB) ou une base de données orientée temps réel qui a des fonctionnalités intéressantes qui pourront servir à optimiser la visualisation
— **Interactivité** : une modularité "acceptable" au niveau des paramètres et des processus surveillés
— **Esthétique** : la visualisation en temps réel se doit d'être lisible, avec des couleurs cohérentes
— **Précision** : la visualisation doit représenter les données sans déformation ou distorsion des informations.

Après avoir recueilli suffisamment d'informations, j'ai pu réaliser une architecture fonctionnelle qui permettrait de répondre à tous ces besoins. Cette architecture est décrite dans la figure 7. J'ai créé à l'occasion un dépôt git accessible à tous (Maurice (2023)), et ai décidé de nommer l'application "EnergyMonitor".

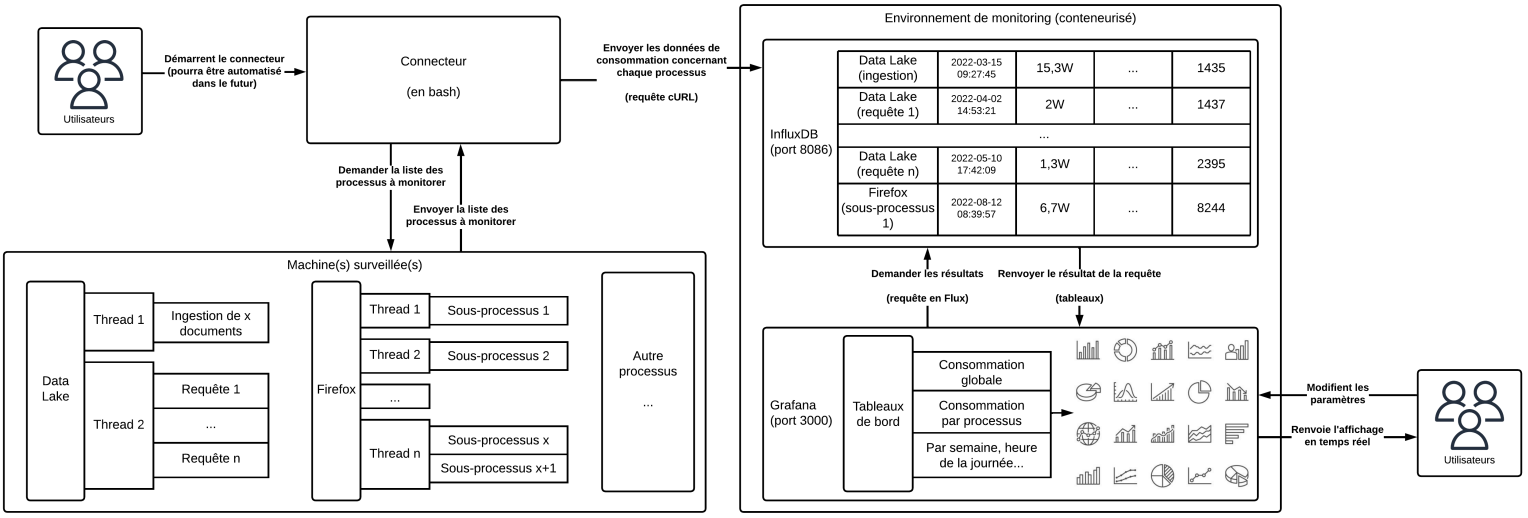

FIGURE 7 – Architecture d'EnergyMonitor

3.2.1.1 Choix des technologies

Pour répondre au premier critère (portabilité), l'idéal était d'utiliser un environnement conteneurisé. L'avantage est qu'il suffit de spécifier certaines images nécessaires au déploiement de notre application pour que cela crée un environnement virtualisé accessible en local, depuis notre machine. J'ai donc choisi Docker pour gérer l'environnement d'exécution. Les conteneurs à déployer concernent, eux, deux domaines de l'application.

Base de données :

- **InfluxDB** : complètement orientée temps réel, elle offre une très faible latence pour l'ingestion et la récupération des données temporelles.
- **Prometheus** : généralement utilisée pour la surveillance et l'alerte. Elle propose énormément de fonctionnalités pour ces domaines. Les fonctionnalités sont presque en tout points similaires à InfluxDB pour le reste, mais sont cependant moins efficaces.
- **TimescaleDB** : base de données flexible et robuste, elle est complètement compatible SQL.
- **SGBD relationnel traditionnel (MySQL, MariaDB...)** : ne sont pas orientées temps réel mais sont extrêmement flexibles au niveau du requêtage et de l'insertion. Permet de gérer le relationnel et donc une complexité sans concurrence par rapport aux autres solutions.

Bien que les bases de données complètement orientées temps réel ne disposent pas d'une flexibilité hors-normes au niveau du requêtage, cela reste la solution la plus viable pour ne pas détruire complètement le critère de performance. De par leur fonctionnement intrinsèque, les SGBD relationnels ne peuvent pas gérer un gros volume de données en temps réel sans compromettre lourdement les performances de la machine.

Dans cette même optique, j'ai donc choisi InfluxDB pour la base de données, même si j'aurais pu utiliser Prometheus car les performances étaient d'un niveau presque similaire (bien qu'Influx soit structurellement beaucoup plus simple, et assure un temps d'exécution moindre, même si la différence est minime). J'ai choisi Influx car je m'étais déjà familiarisé avec le fonctionnement et qu'elle fut mon premier test. TimescaleDB n'offrait pas de performances suffisamment satisfaisantes.

Outil de visualisation :

- **Grafana** : probablement l'outil de visualisation en temps réel le plus populaire, elle dispose d'une connexion aux SGBD très souvent rendue plus facile. L'application est très flexible au niveau des paramètres et de la configuration générale.
- **Kibana** : s'intègre très facilement avec Elasticsearch (décrit dans 3.1.1), il propose des fonctionnalités similaires à Grafana.
- **Plotly** : bibliothèque de visualisation de données qui prend en charge plusieurs langages de programmation, notamment Python. Elle offre des fonctionnalités avancées pour créer des graphiques interactifs et des tableaux de bord en temps réel.

Ici le choix de l'outil était moins crucial que pour le SGBD. Bien que j'étais déjà famillier avec Plotly, j'ai tout de même opté pour Grafana car l'application est plus simple à déployer et que la complexité du niveau de programmation est largement moindre (Plotly induit nécessite de créer un module supplémentaire). De plus, Grafana est réputé pour être plus simple d'utilisation et dispose d'un vaste écosystème. Comme je n'utilisais pas Elasticsearch afin de stocker les données de mesure énergétique, je n'ai pas vu d'intérêt à opter pour Kibana.

3.2.2 Difficultés rencontrées

Le choix d'une TSDB n'a pas facilité la partie "requête" de l'application. En fait, ces bases de données sont **complètement** orientées temps réel, et permettent l'analyse suivie de champs **uniformes**.

J'avais d'abord opté pour une version de InfluxDB plus simple a déployer (InfluxDB 1.8.0-alpine), mais cette version ne permettait pas de faire des regroupements par noms de processus, et n'autorisait même que les **GROUP BY (<temps>)**. Cela était problématique pour de l'analyse de processus groupés. Le langage de requête des versions avant 1.8.0 était le InfluxQL, et ressemblait énormément au SQL traditionnel, bien qu'il était limité sur énormément de

points. J'ai donc décidé de passer sur une version plus récente de InfluxDB, la dernière version stable (2.7.0), qui elle utilise le langage Flux. Les avantages de ce langage sont qu'il est cette fois ci possible de réaliser tous les groupements souhaités, ainsi que de trier les résultats correctement. Le principal désavantage que j'ai trouvé et que l'on ne pouvait pas gérer correctement les listes (pour sélectionner un processus parmi une liste de processus donnée), me forçant donc à réaliser des expressions régulières spécifiques à chaque requête...

Aussi, il n'est pas possible de faire des **requêtes récursives**, pour surveiller par exemple la consommation complète d'un processus donné, comme décrit dans 3.3.3. Il faut connaître à l'avance le niveau de profondeur souhaité pour la récupération des données (ce qui s'est avéré impossible à cause de la disparité de la profondeur des processus). Après quelques recherches, ces défauts sont présents sur toutes les TSDB, mais sont parfois en train d'être étudiés dans les dernières versions.

J'ai imaginé une solution "de secours" qui consisterait à croiser le résultat de requêtes Grafana pour afficher les différentes courbes sur le même graphique, mais cela n'est pas possible non plus car la gestion des variables n'est pas rendue dynamique. La solution parfaite n'existant pas, je travaille sur une solution offrant le meilleur compromis, la complexité se trouvant être directement liée aux programmes utilisés.

3.2.3 Visualisation finale

Le système souffre encore des problèmes précisés dans le paragraphe précédent, bien que je sois en train de procéder à des améliorations générales. La figure 8 montre la surveillance de l'entièreté de la machine. La visualisation comprend trois graphiques, qui constituent un tableau de bord accessible depuis l'accueil de l'application. On peut voir au début de la courbe du graphique en haut à gauche le moment où les pseudo-mesures ont commencées, et à droite le phénomène de décalage de l'obtention d'un pas complet décrit dans 3.3.3.

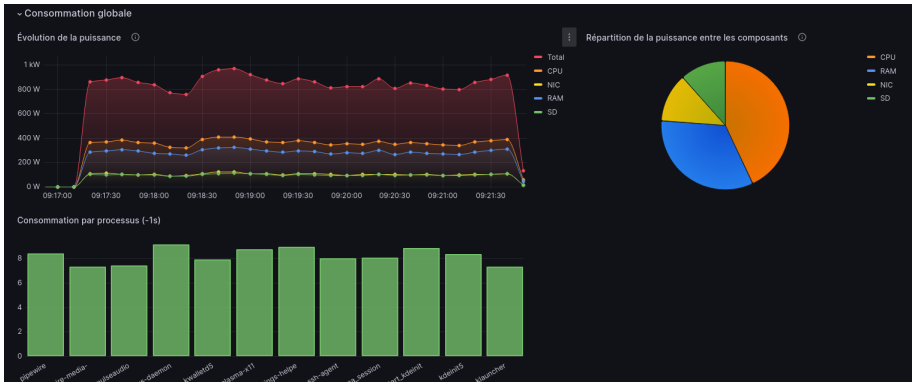

FIGURE 8 – Surveillance de l'entièreté de la machine avec EnergyMonitor

La figure 9 montre la surveillance de l'entièreté de la machine (somme de la consommation des quatre composants) pour un processus en particulier (le cas échéant **systemd**). Une meilleure visualisation permettrait de sélectionner plusieurs processus et de montrer l'évolution de leur consommation sur le même graphique, mais cela est à réaliser en Flux et non en InfluxQL comme décrit dans 3.2.1.1. Pour le moment il est possible d'afficher tous les processus sur cette courbe (ce qui est parfaitement illisible et cause de lourds problème de performances au niveau du navigateur), mais la sélection de processus n'est pas faisable de manière simplifiée (cf. expressions régulières, ce qui n'est pas insurmontable, mais il m'a été demandé de plus me pencher sur le déploiement d'AUDAL vers la fin de mon stage, plutôt que sur la partie visualisation).

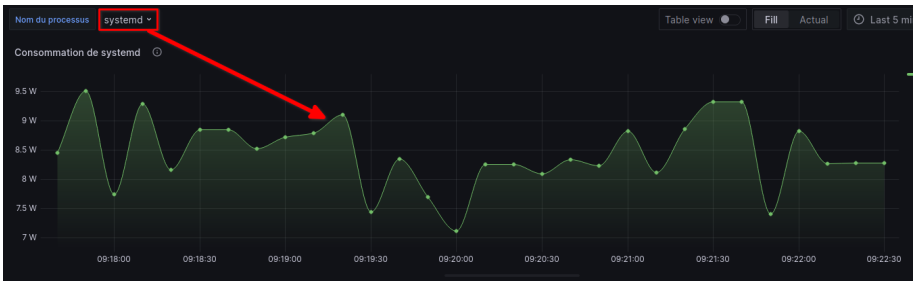

FIGURE 9 – Surveillance du processus **systemd** avec EnergyMonitor

Il faut noter que les valeurs montrées dans les figures 8 et 9 sont des valeurs aléatoires générées par mes scripts, uniquement dans un but de démonstration de ma solution, et ne rendent pas compte de la consommation réelle des processus et encore moins de la machine au global. Cela n'invalide pas la solution développée, et se justifie en fonction du temps de développement des scripts de mesures de mon tuteur, précisé dans 3.3.3, induisant un retard dans l'obtention de valeurs fiables et ingérables dans la visualisation en temps réel.

3.3 Scripts de mesure énergétique

Pour les besoins de notre étude, nous avions besoin d'utiliser d'autres logiciels qui nous permettraient de comparer nos résultats de consommation et d'avoir une base, en particulier pour le CPU. Ces logiciels sont moins précis que nos mesures (présentées dans la section 3.3.3) du fait du nombre de composants mesurés moindres, mais aussi des interfaces utilisées.

3.3.1 Déploiement de PowerJoular

L'intitulé de la mission de stage ne parlait que de PowerJoular, un projet permettant de mesurer la consommation énergétique sur des ordinateurs disposant d'un processeur compatible avec la technologie RAPL (Running Average Power Limit) de Intel, une fonctionnalité de gestion de l'alimentation fournie par les processeurs Intel, et sous Raspberry Pi. Le projet est porté dans l'étude Noureddine (2022b) et dispose lui aussi de son dépôt git (Noureddine (2022a)).

Difficultés rencontrées Bien que l'installation se soit déroulée sans soucis, nous avons rencontrés des problèmes lors de l'analyse des mesures obtenues.

- **GPU** : cette mesure est décrite de manière extrêmement floue dans la documentation. Après une analyse complète du code permettant d'obtenir cette figure ainsi qu'une confirmation de la part de l'auteur, cette mesure n'est significative en rien par rapport à un processus en particulier.
- **Résultats obtenus** : les résultats obtenus nous semblaient complètement indifférents de la charge induite par les processus. Nous n'avons pas réussi à déterminer de si cela venait de notre environnement, de notre configuration, ou d'un problème de PowerJoular directement, même après une discussion avec l'auteur.

Ces incertitudes se sont révélées être suffisamment importantes pour que nous prenions la décision de ne plus utiliser PowerJoular pour nos mesures.

3.3.2 Déploiement de Scaphandre

Scaphandre est une solution largement plus industrialisée que PowerJoular. À la différence de cette dernière qui est développée en Ada, Scaphandre est développée en Rust, un langage de programmation puissant et léger. Les communautés développant en Rust sont réputées pour maintenir de manière très efficace leur code, tout en respectant beaucoup de principes fondamentaux de la programmation. De ce fait, il n'a pas été compliqué de déployer Scaphandre sur n'importe quel environnement. Scaphandre dispose également d'un dépôt git (Hubblo (2022)).

Cependant, les difficultés rencontrées proviennent de l'objectif dans lequel Scaphandre a été développé. Scaphandre est utilisé pour gérer la consommation globale de l'ordinateur, puisqu'il est destiné à des environnements déployés dans un cluster, typiquement sous Kubernetes. La conséquence directe est que l'analyse par PID n'est pas développée de manière aussi approfondie qu'on le

souhaiterait. De même, les résultats sont très approximatifs (même s'ils faisaient plus de sens que ceux donnés par PowerJoular).

Difficultés rencontrées

- **"System-wide"** : Scaphandre est un outil est orienté "System-wide", c'est à dire qu'il est fait initialement pour mesurer tout le système. Cela nous a contraint à trouver une solution grossière qui consiste à lancer Scaphandre en parallèle de nos expériences de mesures de consommation, et de récupérer après le déroulement complet les valeurs pour les processus qui nous intéressent
- **Sortie de fichiers** : la méthode de sortie de Scaphandre ne proposait pas, en spécifiant un processus en particulier, de retourner la sortie quand le processus n'existait plus. Nous avons donc du mesurer l'ensemble de la machine et renvoyer la sortie dans un fichier JSON, en enregistrant en parallèle les processus qui nous intéressaient, pour pouvoir les extraire du fichier plus tard. Après quinze heures d'exécution, le fichier renvoyé faisait 130mo, ce qui n'est pas tant en soi. Cependant, un tel fichier demande à un programme Python plus de 40 minutes de traitement à cause de sa structure malformée, qui obligeait à faire une recherche en profondeur de chacun des processus.

Les valeurs de Scaphandre étaient tout de même bien plus élevées que les valeurs des scripts de mon tuteur, mais respectaient la proportion par rapport à la charge du processeur, ce qui n'était pas le cas pour PowerJoular.

3.3.3 Validation des scripts de mon tuteur

Il faut noter qu'en parallèle de ces activités, mon tuteur de stage réalisait et améliorait ses scripts permettant les mesures énergétiques. Nous étions alors dans une approche expérimentale.

La première chose à dire est la suivante : "Nous sommes dans l'incapacité au sens fort de fournir des données de consommation énergétique qui ne sont pas des approximations si nous utilisons des programmes informatiques". En effet, il est impossible d'obtenir une telle mesure sans utiliser de capteur physique, accolé à chaque composant matériel. Dès lors, nous fournissons des approximations (aussi précises et cohérentes soient-elles) basées sur des formules mathématiques accédant à des informations stockées par l'ordinateur, dont l'accès est favorisé par l'utilisation de systèmes sous Linux, grâce au Linux File System (Linux FS). Ces informations sont situées dans **/proc/<PID>/stat** et **/proc/<PID>/task/<TID>/stat**. Les fichiers **/proc/*** sont des fichiers dits "virtuels" en que ceux-ci ne résident pas sur le disque sous une forme traditionnelle. Ils sont plutôt générés à la volée par le noyau du système lorsque vous y accédez, dans la RAM. Cela permet d'obtenir des informations en temps réel sur les processus, les périphériques, les ressources système, etc.

Nous avons, lors de la rédaction de ces scripts, atteint une limite qui est celle du fonctionnement normal d'un ordinateur. La suite de cette section servira à vulgariser le plus simplement possible les limites atteintes.

L'accès aux informations concernant les processus Chaque opération coûte un temps incompressible à un ordinateur, du fait de son fonctionnement. Nous avons la possibilité d'accéder à des informations concernant un processus donné grâce à son identifiant (*Process ID*, PID en anglais) où à ses *threads* (fils d'exécution en français, et sont eux caractérisés par un *Thread ID*, TID), telles que la quantité de temps que le processeur a alloué à ce PID sur la dernière seconde par exemple. Cette information peut nous donner une idée du niveau de "stress" qu'un processus a apporté, et donc, de la proportion de la consommation maximale du processeur sur cette période de temps donnée. Le soucis est que l'accès à ces informations, bien que facilité par le Linux FS, reste non-négligeable, et la complexité est démultipliée quand il faut regarder aussi les *threads* du processus.

Les processus enfants Imaginons que l'ordinateur est comme une grosse usine où plein de petites tâches sont effectuées. Mais attention, ces tâches ne sont pas toutes faites en même temps, c'est comme si chaque tâche attendait son tour pour être réalisée. Chaque tâche s'appelle un processus, et ils sont gérés par le processeur de l'ordinateur. Le processeur donne à chaque processus un certain moment pour travailler sur une partie de sa tâche. Prenons l'exemple de Firefox, le navigateur internet. Il constituera le processus A. Quand il lui est demandé de charger une page web, il va créer un sous-processus B pour faire cela. Si la prochaine tâche constitue à regarder une vidéo, Firefox peut créer un autre sous-processus C pour lire cette vidéo. Maintenant, parlons de la consommation d'énergie. Si on veut mesurer combien d'énergie Firefox utilise, on doit prendre en compte non seulement Firefox lui-même, mais aussi tous ses petits processus enfants, comme B et C. L'existence de ces sous-processus est montrée dans la figure 10, avec en vert les processus parents, et en rouge, tous leurs enfants. Il est important de noter qu'il peut y avoir des petits-enfants et ainsi de suite, nécessitant un traitement récursif extrêmement gourmand.

FIGURE 10 – Extrait du résultat de la commande **pstree** sous Linux

Certains problèmes peuvent occurrer : parfois, lorsque l'on mesure la consommation d'énergie, il peut y avoir des valeurs manquantes. Cela se produit parce que nous devons accéder aux informations de consommation d'énergie dans des fichiers spéciaux du système, et parfois cela peut prendre du temps. Il faut donc éviter ce phénomène en augmentant la fenêtre de temps. La contrepartie sera d'avoir des valeurs moins exhaustives, et causera un "décalage" ou délai d'accès aux données complètes.

La figure 11 montre graphiquement les données obtenues par l'algorithme permettant de récupérer les données de consommation pour le processus **Firefox** ainsi que de tous ces enfants. L'axe des abscisses représente alors le temps que prend notre étape à être réalisée. Une étape constitue à d'abord prendre le processus passé en paramètres (le cas échéant Firefox) puis a analyser tous les processus enfants de ce dernier. Les valeurs des deux axes des sont purement arbitraires et ne représentent pas la réelle consommation du navigateur web, ni le temps de complétude de notre étape.

On rappelle qu'il faut, pour "dézoomer" dans le diagramme et avoir l'évolution de la consommation de Firefox, faire la somme des processus enfants de Firefox. Nous dépendons alors du temps de chaque étape, qui varie en fonction des performances de la machine (légèrement) et du nombre de sous-processus (énormément). Cette somme est illustrée dans la figure 12. La barre surlignée en rouge est la valeur de consommation réelle de Firefox et constitue la mesure finale.

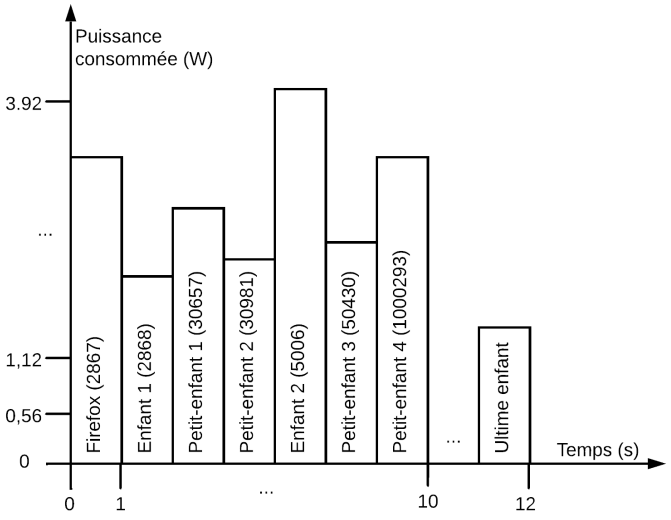

FIGURE 11 – Schématisation de la puissance utilisée par Firefox (à l'instant T0) puis par ses sous processus (à un instant Tx)

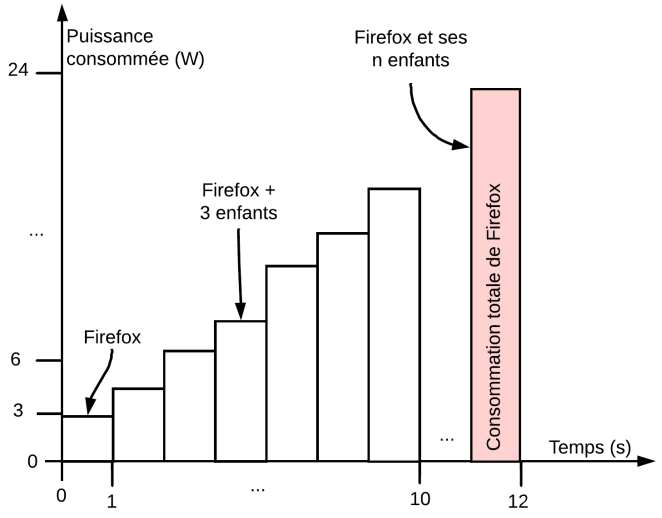

FIGURE 12 – Schématisation de la somme successive de la puissance utilisée par Firefox et de ses processus sur un intervalle de temps

Les processus extrêmement courts Une seconde problématique est celle de la durée des processus. Un traitement unique et long est en général simple à mesurer. Si celui là appelle des enfants, la charge et la complexité du système à développer est plus importante. Enfin, si ces processus enfant sont extrêmement courts, cela laisse peu de place à nos programmes de mesure énergétique de pouvoir effectuer correctement leur tâche, car la fenêtre de temps pour les capturer est extrêmement faible. Cette notion de fenêtre de temps est illustrée dans la figure 13. En général, les opérations effectuées par un Data Lake pour ingérer les différentes données dans la structure sont petites, et consistent souvent en un très court processus pour ingérer un document.

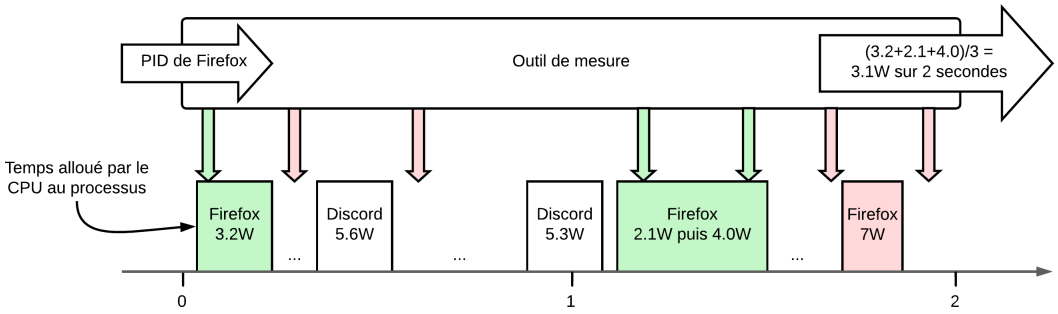

FIGURE 13 – Schématisation du temps CPU alloué à chaque processus et de la tentative de lecture des informations de consommation en parallèle

3.3.4 Conclusion

La table 2 illustre les avantages et inconvénients de chaque outil selon un critère spécifique. Une croix (X) sur fond vert indique que le critère est complètement validé, un tilde (∼) sur fond orange indique que le critère est validé sous certaines conditions, et une barre oblique (/) sur fond rouge indique que le critère n'est pas validé, sous aucune circonstance. Ces critères ont été choisis par rapport aux besoins de notre étude, et les valeurs qui sont précisées sont des valeurs relativement arbitraires afin de ne pas rentrer dans des détails inutiles.

Voici le détail de ces critères :
- Facile à déployer : on peut installer l'application sans avoir à toucher au code
- Capte les processus courts : un processus court est ici défini comme durant moins de 200 millisecondes
- Mesures "complètes" : donnent des mesures complètes les applications qui peuvent donner les mesures de consommation du CPU, la RAM, la carte réseau et des appareils de stockage
- Mesures cohérentes : la relation entre le niveau de stress d'un composant et la valeur obtenue est proportionnelle
- Processus spécifiques : il est possible de ne mesurer qu'un processus spécifique, précisé à l'avance
- Fréquence de mesure minimum correcte : est considérée comme correcte une fréquence permettant d'obtenir des valeurs sans augmenter le pas minimum (illustré dans la figure 12) de validation des mesures pour un processus spécifique

	Facile à déployer	Capte les processus courts	Mesures "complètes"	Mesures cohérentes	Processus spécifiques	Fréquence de mesure minimum correcte
PowerJoular	X	X	/	/	X	X
Scaphandre	X	/	/	∼	/	∼
Domolandes	∼	X	X	X	X	∼

Table 2 – Avantages et inconvénients de chaque outil de mesure

4 Bilans

4.1 Bilan général

J'ai réellement apprécié réaliser ce stage chez Domolandes, tout particulièrement car il m'a permis de découvrir le monde de la recherche autrement qu'en surface. J'ai pu me renseigner pleinement sur le fonctionnement des recherches, par rapport aux publications, aux présentations en conférences et aux échanges avec d'autres chercheurs. Cette immersion dans le processus de recherche m'a donné une perspective plus approfondie et m'a permis de comprendre les défis et les exigences de ce domaine passionnant.

Comme précisé dans la section 1.1, je me considérais comme étant moins à l'aise dans tout ce qui concernait la programmation système et la compréhension de ces derniers. Après ce stage, j'estime que mon niveau dans ce domaine a grandement augmenté, et c'est pour moi une petite fierté que d'avoir considéré réaliser mon stage dans ce domaine.

4.2 Compétences du B.U.T.

- **C1 - Réaliser un développement d'application** : Il m'a fallu développer une application de mesure de la consommation énergétique et réaliser les scripts d'automatisation du déploiement, en organisant le système de fichier pour pouvoir récupérer et nettoyer les données après le déroulement des expériences. Aussi, la compréhension et l'analyse des applications réutilisées demandent beaucoup de compétences dans le développement en général, et touchent évidemment cette compétence.
- **C2 - Optimiser des applications** : Il a fallu travailler en faisant extrêmement attention à la consommation et la concurrence entre les processus. Afin de ne pas entraver l'objectif premier des expériences, mon application de surveillance en temps réel respecte des contraintes de performance très strictes puisque les *benchmarks* sont demandeurs en ressources matérielles, et une application supplémentaire qui demande à surveiller chaque processus impose très rapidement une charge importante à la machine. De plus, tous les scripts développés ont été fait en veillant à ce que les opérations effectuées ne demandent pas de traitement inutiles au système.
- **C3 - Administrer des systèmes informatiques communicants complexes** : Les bases de données utilisées communiquent sans arrêt sur des ports qu'il a fallu configurer. De plus, une bonne partie des traitements étudiés, spécifiquement dans le domaine des Data Lakes, sont orientés réseau, avec des extractions d'archives web, documents etc., aussi bien au niveau réception qu'envoi. Il faut notamment surveiller des processus réalisant des requêtes vers le web, et nous avons réalisé beaucoup d'expériences pour essayer de les capter au mieux (bien que cet aspect relève plus de l'étude des systèmes que du réseau en soi).

— **C4 - Gérer des données de l'information** : Cette compétence est au cœur du sujet du stage et même du domaine d'étude du laboratoire. Proposer une étude sur un Data Lake nécessite de comprendre le fonctionnement des données que l'on traite, leur architecture. Bien que le B.U.T. propose en 2e année une approche approfondie sur plusieurs des technologies utilisées par notre Data Lake, nous n'allons guère plus loin dans le non-relationnel. La partie sur la visualisation en direct de la consommation sollicite aussi extrêmement ce pôle de compétences car j'utilise aussi un SGBD particulièrement optimisé pour le temps réel comme InfluxDB, et ai du réaliser les requêtes adaptées pour afficher correctement les graphiques.
— **C5 - Conduire un projet** : La conduite du projet est complètement conditionnée par le pipeline d'évolution de l'étude. L'idée est de publier un article qui sera présenté lors d'une conférence nationale ou internationale, imposant ainsi des contraintes de temps (bien que ce ne soit pas que ça), tout en continuant en parallèle l'étude au mieux vers l'objectif final. Il m'a fallu comprendre les enjeux et essayer d'avoir une vision plus globale pour proposer un niveau que je juge correct selon mes connaissances actuelles.
— **C6 - Collaborer au sein d'une équipe informatique** : Tout au long du stage il y eut des réunions, concertations avec les autres membres de l'équipe. Bien que le changement de stratégie employé vers la fin de mon stage (privilégier la complétude du système étudié plutôt que de se précipiter pour avoir des valeurs concernant les expériences) ait changé la fréquence de ces réunions, elles n'en sont pas moins restées presque quotidiennes au début du stage, imposant des jalons relativement fréquents (bien que souvent non atteints...). Pour améliorer la cohésion d'équipe, nous avons tenté de faciliter la collaboration en appliquant un principe de la gestion de projet. Bien que nous ayons mis en place un Trello (tableau kanban), il faut admettre que l'activité de celui-ci n'a pas été assurée tout au long du stage. Je ne dirais pas que c'est un manquement majeur, mais cela reste définitivement quelque chose sur lequel il me faut travailler.

4.3 Perspectives professionnelles

Comme je l'ai déjà mentionné de multiples fois dans ce rapport, la recherche est réellement un domaine qui m'attire. En plus d'avoir eu la possibilité de me renseigner énormément sur le sujet, je peux maintenant me projeter plus aisément dans mon avenir professionnel. Beaucoup de personnes m'avaient avancé que la recherche était un domaine compliqué dans lequel il était parfois compliqué de s'épanouir, mais je ne pense pas que cette voie me déplaise particulièrement. Après cette expérience, j'envisage même de continuer dans le domaine de la R&D.

Domolandes a évoqué à de multiples reprises la possibilité d'effectuer un contrat en alternance, et si cela se concrétise, c'est avec grand plaisir que je considérerai leur offre. L'alternance offre de nombreux avantages, notamment la possibilité d'acquérir une expérience professionnelle tout en poursuivant ses études. Cela me permettrait d'approfondir mes connaissances théoriques tout en les appliquant concrètement dans un environnement de recherche. De plus, travailler en alternance avec une entreprise comme Domolandes me permettrait de bénéficier de leur expertise et de collaborer avec des professionnels expérimentés, ce qui serait extrêmement enrichissant pour ma formation.

Ma voie dans l'informatique n'a jamais été tracée depuis le départ, et c'est un petit peu par hasard si je me suis orienté complètement dans le domaine. Pour ce qui est de ma relation avec cette discipline, je suis généralement quelqu'un qui n'apprécie pas les artifices, et suis souvent intransigeant concernant la qualité des systèmes articulés que l'on manipule. Travailler à un niveau nécessitant des connaissances techniques approfondies ne peut être, à mon avis, que bénéfique pour moi, et cela peut me permettre de m'émanciper intellectuellement. J'apprécie énormément travailler sur des sujets qui concernent ce que je considère comme l'essence de l'informatique et qui sont la raison de pourquoi j'étudie actuellement dans ce domaine.

Références

Alvarez Valera, H. H. (2022). *An energy saving perspective for distributed environments : Deployment, scheduling and simulation with multidimensional entities for Software and Hardware*. PhD thesis, Université de Pau et des Pays de l'Adour. Thèse de doctorat dirigée par Dalmau, Marc et Roose, Philippe Informatique Pau 2022.

Hubblo (2022). Scaphandre. `https://github.com/hubblo-org/scaphandre`. Energy consumption metrology agent.

Maurice, A. (2023). EnergyMonitor. `https://github.com/Hizaak/EnergyMonitor`. Visualiser l'évolution en temps réel de la consommation énergétique de processus spéficiques.

Noureddine, A. (2022a). PowerJoular. `https://github.com/joular/powerjoular`. PowerJoular allows monitoring power consumption of multiple platforms and processes.

Noureddine, A. (2022b). PowerJoular and JoularJX : Multi-Platform Software Power Monitoring Tools. In *18th International Conference on Intelligent Environments*, Biarritz, France.

Orogat, A. (2021). Data lakes for big data [report on existing data lakes].

Sawadogo, P. (2021a). AUDAL. `https://github.com/Pegdwende44/AUDAL`.

Sawadogo, P. (2021b). AudalMetadata. `https://github.com/Pegdwende44/AudalMetadata`.

Sawadogo, P. N. and Darmont, J. (2021). Benchmarking data lakes featuring structured and unstructured data with DLBench. In *Big Data Analytics and Knowledge Discovery*, pages 15–26. Springer International Publishing.

Sawadogo, P. N., Darmont, J., and Noûs, C. (2021). Joint Management and Analysis of Textual Documents and Tabular Data within the AUDAL Data Lake. In *25th European Conference on Advances in Databases and Information Systems (ADBIS 2021)*, volume 12843 of *Lecture Notes in Computer Science*, pages 88–101, Tartu, Estonia. Springer.