RAPPORT DE STAGE

Licence 3 Informatique

le sujet

Le fonctionnement d'un centre de données sur les énergies renouvelables

rédigé par

Ishak Ayad

Juin 2019

Responsable entreprise :**Marwa Dammak**

Responsable UCP :**Iryna Andriyanova**

Durée du stage : 4 mois du 23 avril au 22 août 2019

Remerciements

Tout d'abord, je tiens à remercier le personnel de l'ENSEA pour m'avoir accueilli pendant ce stage.

Mes remerciements vont à mon tuteur Mme.Iryna Andriyanova pour m'avoir permis de faire ce stage dans un environnement chaleureux et les meilleures conditions possibles, d'avoir pris son temps pour m'expliquer les concepts théoriques et expérimentaux cruciaux à la compréhension de ce stage, ainsi que le fonctionnement d'un laboratoire de recherches et toutes les formalités qui permettent à un laboratoire de fonctionner correctement.

Aussi, je remercie mon maître de stage Mme. Marwa Dammak qui m'a formé et accompagné tout au long de cette expérience professionnelle avec beaucoup de patience et de pédagogie. Je tiens à la remercier également pour son aide précieuse, son soutien constant et ses encouragements qui m'ont bien permis de mener à bien mes recherches, et dont la gentillesse et la sympathie ont ajouté à ce stage une touche fortement agréable.

De plus je souhaite faire part de ma reconnaissance au post-doctorant Muhammad Ali pour m'avoir éclairé sur les concepts de la virtualisation et pour avoir répondu à mes questions.

Je veux aussi remercier Mr.Lemaiare pour ses explications détaillées sur le fonctionnement du serveur web devweb.etu et Mr.Laroque pour m'avoir éclairé sur le fonctionnement des commandes de calcul du taux d'usage des CPU et du fichier /proc/stat.

D'autre part, je tiens aussi à remercier Mr.Laurent Protois qui m'a beaucoup aidé sur l'installation des outils (iDRAC, ipmitool, ...) sur les serveurs.

Enfin, je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage : ma famille et mes camarades de promotion.

Résumé et abstract

Résumé

Depuis plusieurs années, La consommation des data centers à la base du réseau internet ne cesse de croître, au point de représenter 4 % de la consommation énergétique mondiale en 2015. La climatisation et les systèmes de refroidissement représentent de 40 à 50 % de la consommation énergétique des data centers. Les data centers américains ont consommé 91 milliards de kWh en 2013 et 56 milliards en Europe (prévision : 104 milliards en 2020).

Dans la lignée des travaux du projet **ECOBIO H2** et en association avec L'**ENSEA**, qui ont permis d'obtenir un centre de données et un laboratoire, mon travail a consisté, en premier lieu à effectuer des tests de consommation et des mesures sur un prototype de centre de données.

Le but principal du projet est l'étude du fonctionnement d'un centre de données sur les énergies renouvelables. Cela en effectuant des études de différents matériaux ainsi en choisissant les équipements qui convient à chacun. Parmi ces derniers, il s'agit principalement des serveurs. Ainsi l'objectif est d'étudier leurs performances en terme de capacité et de la consommation énergétique et de faire une comparaison.

Mots clés :CPU, consommation énergétique, architecture tiers 3, serveur web, script bash, iDRAC

Abstract

For several years, the consumption of data centers at the base of the Internet network continues to grow, to the point of representing 4 % of global energy consumption in 2015. Air conditioning and cooling systems account for 40 to 50 % of the energy consumption of data centers. US data centers consumed 91 billion kWh in 2013 and 56 billion in Europe (forecast : 104 billion in 2020).

In line with the work of the project **ECOBIO H2** and in association with **ENSEA**, which made it possible to obtain a data center and a laboratory, my job consisted in first carrying out tests consumption and measurements on a prototype data center.

The main purpose of the project is to study the operation of a renewable energy data center. This by conducting studies of different materials as well as choosing the equipment that suits everyone. Of these, they are mainly servers. To study their performance in terms of capacity and energy consumption and to make a comparison.

Keywords :CPU, energy consumption, third-party architecture 3, Web server, script bash, iDRAC

Table des matières

Table des figures

Listings

Introduction

Du 23 Avril jusqu'à maintenant, j'ai effectué un stage au sein du laboratoire ETIS (située à l'École nationale supérieure de l'électronique et de ses applications.),Cergy. Au cours de ce stage avec l'équipe de Télécommunications, j'ai pu m'intéresser au fonctionnement d'un centre de données sur les énergies renouvelables.

Plus largement, ce stage a été l'opportunité pour moi d'appréhender la gestion de projet dans une équipe de recherche, ainsi découvrir le milieu de la recherche. Cela en apprenant à faire des bibliographies et des recherches. J'ai aussi bénéficié d'une expérience sur les centres de données et les outils utilisés pour gérer ce genre de matériel.

Au-delà d'enrichir mes connaissances en réseaux et télécommunication, ce stage m'a permis de comprendre dans quelle mesure que mon parcours professionnel pourrai basculer vers de la recherche.

Mon stage au sein de l'équipe de Télécommunications a consisté essentiellement en le fonctionnement d'un micro datacenter sur les énergies renouvelables.

Mon maître de stage étant ATER (Attaché temporaire d'enseignement et de recherche), j'ai pu apprendre dans d'excellentes conditions comment gérer un projet professionnel, comment rédiger des bibliographie ...etc

Ce stage a donc été une opportunité pour moi de percevoir comment un laboratoire dans un secteur de la Télécommunication (ICI). Les activités de recherche de l'équipe ICI s'articulent autour de quatres axes :

— Codage correcteur d'erreurs
— Théorie de l'information
— Allocation des ressources
— Imagerie

L'élaboration de ce rapport a pour principale source les différents enseignements tirés de la pratique journalière des tâches auxquelles j'étais affecté. Enfin, les différents tests que j'ai pu effectué sur le micro centre de données m'ont permis de donner une cohérence à ce rapport.

Pour rendre compte de manière fidèle et analytique des deux mois passés au sein du laboratoire ETIS avec l'équipe ICI, il apparaît logique de présenter à titre préalable les résultats que j'ai eu, à savoir la présentation du Projet (I), puis d'envisager le cadre du stage : le laboratoire ETIS, tant d'un point de vue sur le travail réalisé (II). Enfin, il sera précisé les différentes missions et tâches que j'ai pu effectuer au sein du laboratoire et les nombreux apports que j'ai pu en tirer(III).

Chapitre 1

Le projet

1.1 Présentation

Chapeau Dans cette section, je présente le lieu du stage, l'équipe, le sujet et thème et projet du stage, et enfin l'environnement de travail avec les outils utilisés.

1.1.1 ETIS | ENSEA

ETIS, laboratoire des Équipes Traitement de l'Information et Systèmes, est une équipe de recherche commune à l'ENSEA et à l'Université de Cergy-Pontoise, reconnue par le CNRS (UMR 8051).

ETIS est organisé en quatre équipes de recherches, repartis sur les sites de l'ENSEA, et de l'Université de Cergy-Pontoise (St-Martin 1) :

1. **Architecture, Systèmes, Technologies pour les unités Reconfigurables Embarquées (ASTRE) :**
   — Adéquation Algorithme-Architecture, Conception d'IP
   — Systèmes sur puces reconfigurables (RSoC)
2. **Information, Communications, Imagerie (ICI) :**
   — Algèbre linéaire
   — Imagerie
   — Codes LDPC et récepteurs itératifs
   — Allocation de ressources pour la couche physique
3. **Indexation Multimédia et Intégration de Données (MIDI) :**
   — Indexation multimédia
   — Intégration de données
4. **Neurocybernétique :**
   — Perception visuelle
   — Apprentissage et imitation
   — Navigation et planification
   — Systèmes complexes

L'ENSEA est l'École nationale supérieure de l'électronique et de ses applications.

1.1.2 L'ÉQUIPE

ETIS regroupe environ 80 membres, dont la moitié sont des enseignants-chercheurs (ENSEA et Université de Cergy-Pontoise).

L'équipe de ce projet est composée principalement de :

1. Équipe ETIS :
   — Iryna Andriyanova — Maîtresse de conférences
   — Pierre Andry — Maître de conférences
   — Inbar Fijalkow — Professeur des université
   — Marwa Dammak — ATER
   — Muhammad Ali — Doctorant
   — Louis Desportes — Doctorant
2. Ecobio :
   — Jérôme David — Contact Technique

1.1.3 SUJET DE RECHERCHES

Notre équipe se concentre sur la partie du un micro datacenter dont les objectifs sont :

► Offrir une solution d'hébergement de données numériques.

► L'étude du fonctionnement d'un centre de données sur les énergies renouvelables.

► Étudier les performances des serveurs en terme de capacité et de la consommation énergétique.

1.1.4 THÈME DE STAGE

► Le fonctionnement d'un centre de données sur les énergies renouvelables.

L'objectif de mon stage est d'effectuer des études de différents matériaux ainsi en choisissant les équipements qui convient à chacun, principalement il s'agit de serveurs.Ce afin de leurs associer des tests pour récupérer des statistiques. Puis étudier leurs performances en terme de capacité et de la consommation énergétique et faire une comparaison.

1.1.5 PROJETS

Le projet de démonstrateur ECOBIOH2 s'inscrit dans un programme immobilier d'éco-îlot, nommé ECOBIO. Ce programme consiste à construire, au cœur de ville d'Avignon, sur le site actuel du magasin Biocoop, un bâtiment résidentiel-tertiaire neuf de 10 000 m², à haute performance environnementale. Cet îlot regroupe plusieurs activités : commerces, restauration, bureaux, hébergement, activités culturelles, ferme urbaine, parking ... La haute performance environnementale - certifiée par les labels Bâtiment Durable Méditerranéen (label BDM) et E+/C- niveau E4/C2 - s'appuie en particulier sur :

— une conception bioclimatique biossourcées ;
— une production solaire associée à un stockage d'électricité hybridé (hydrogène + batterie) intégré au bâtiment ;
— une gestion des flux intelligente d'un point de vue environnemental (énergie, eau, chaleur, ventilation, déchets organiques).

Sur la base de ce programme, le projet ECOBIO H2 consiste à intégrer au bâtiment un système complet associant : un stockage d'électricité hybridé (batterie + chaîne hydrogène), un micro datacenter et un logiciel auto-apprenant. Les objectifs de ce système complet intégré à l'éco-îlot sont :

— D'offrir une solution d'hébergement de données numériques innovante, locale et 100% renouvelable.

— D'augmenter les performances d'autoconsommation de l'énergie solaire produite par le bâtiment, grâce à l'intégration d'un stockage hybridé (batterie + chaîne hydrogène) et à un pilotage auto apprenant favorisant la maîtrise des consommations.

1.1.6 ENVIRONNEMENT DE TRAVAIL, OUTILS UTILISÉS

► **Système d'exploitation** Les serveurs sont configurer avec un système d'exploitation ubuntu 19.04, pour s'y connecter j'utilise source shell (SSH) qui est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient donc impossible d'utiliser un sniffer pour voir ce que fait l'utilisateur.

Ainsi, avec un système d'exploitation ubuntu on a accès aux différents fichiers de log ubuntu (/var/log/messages, /proc/stat, /proc/-meminfo, ... etc).

► **IPMI** L'IPMI, pour "Intelligent Platform Management Interface", est une interface de gestion de matériels standardisés, fournie principalement sur les serveurs (Dell, IBM, HP, Intel, NEC, Supermicro, etc), indépendante du système d'exploitation, et destinée à contrôler certains composants hardware (sondes de températures, vitesses de rotation des ventilateurs, etc) et à manager la machine, localement ou à distance et quelque soit son état (éteinte ou allumée).

► **Les serveurs** On dispose de deux serveurs Dell :

☐ **Serveur dioxine** : c'est un serveur rack haute densité 1U extensible avec une conception monosocket efficace pour un faible coût total de possession. Idéal pour la virtualisation dense et le stockage SDS extensible 32C.

☐ **Serveur glyphosate** : smart Value PowerEdge R630 Server pour optimiser l'efficacité du datacenter avec un moteur de base de données ou de virtualisation ultradense qui prend en charge jusqu'à 10 disques SSD Flash dans un boîtier 1U 16C.

► **iDRAC** iDRAC est un contrôleur (integrated Dell Remote Access Controller), il aide a déployer, mettre à jour, surveiller et entretenir les serveurs Dell powerEdge avec ou sans agent logiciel de gestion des systèmes.

1.1.7 VUE DE L'ENSEMBLE

Les serveurs sont placés dans une baie [1] au niveau de la salle des serveurs [2] du laboratoire.

FIGURE 1.1 – baie des serveurs

FIGURE 1.2 – disposition des serveurs sur la baie

1. Une baie (en anglais rack, râtelier) est une armoire très souvent métallique parfois à tiroirs mais généralement à glissières (ou rails) destinée à recevoir les boîtiers d'appareils, généralement électroniques, réseaux ou informatiques de taille normalisée.

2. Une salle de serveur est une salle, généralement climatisée, consacrée au fonctionnement continu de serveurs informatiques

L'architecture des serveurs est une architecture tiers (III) ou dite l'architecture à trois niveaux.

1.1.8 Présentation de l'architecture à trois niveaux

Un réseau hors de contrôle : Avant de pouvoir identifier un bon réseau, vous devez identifier un mauvais réseau. Malheureusement, le réseau en est rempli et vous allez y faire face quotidiennement. Souvent, les petites et moyennes entreprises sous-estiment l'importance d'un réseau sain. En conséquence, ils n'engagent pas de spécialiste réseau pour leur réseau. Cela ne peut signifier qu'un réseau ne répondant pas aux trois exigences définies ci-dessus. Voici un exemple.

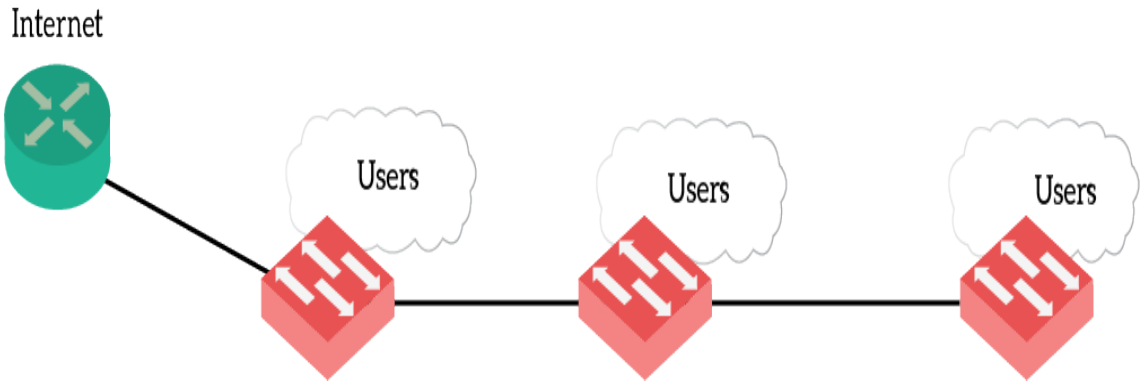


FIGURE 1.3 – Votre réseau ne devrait jamais ressembler à ceci

C'est l'exemple classique d'un réseau horrible. En jargon, il s'agit d'une chaîne de commutateurs : une série de commutateurs connectés les uns à côté des autres. Quelque chose que nous devons éviter. Même dans ce cas, il s'agit du déploiement le plus courant que vous allez trouver dans des réseaux organisés. La raison en est simple. Au tout début, votre routeur se connecte à Internet et un commutateur permet de connecter tous les utilisateurs. À ce stade, le réseau n'est pas si mauvais pour le moment. Cependant, à mesure que le nombre d'utilisateurs augmente, votre commutateur ne peut pas tous les accueillir. À ce stade, le «responsable informatique» apparaît et connecte un nouveau commutateur pour desservir davantage d'utilisateurs. L'activité continue de croître et le réseau ne peut pas prendre en charge davantage d'utilisateurs. Ensuite, quelqu'un ajoute un autre commutateur à la chaîne.

L'architecture à trois niveaux Également appelé modèle hiérarchique à trois couches, il s'agit de la conception phare de Cisco pour les réseaux Campus. Ses concepts peuvent toutefois être appliqués à n'importe quel réseau, y compris les centres de données. Cette architecture à trois niveaux est la solution la plus évolutive et résiliente. Sa conception claire et simple le rend également très facile à gérer. La combinaison de ces caractéristiques fait de cette conception la solution idéale pour les grandes entreprises, où vous devez connecter des milliers d'appareils sur un grand campus, même sur plusieurs bâtiments.

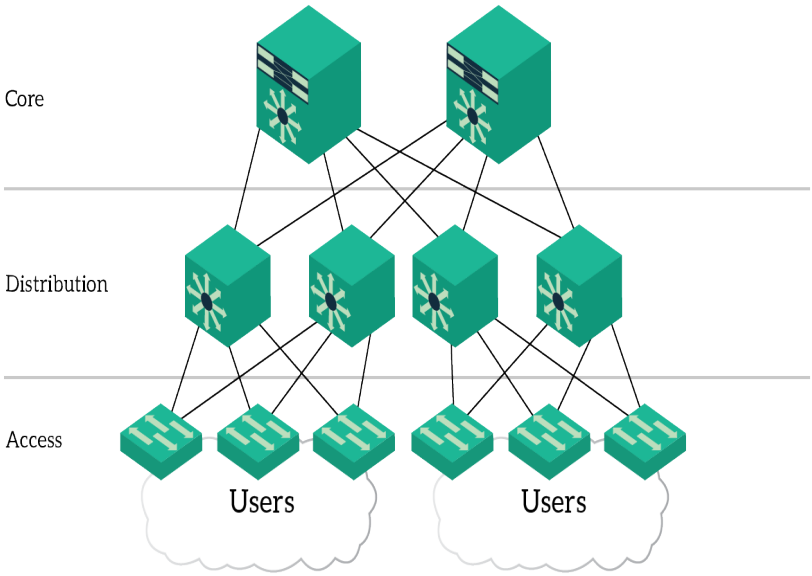


FIGURE 1.4 – Architecture à trois niveaux selon Cisco

— **La couche d'accès :**La couche d'accès est celle la plus proche des utilisateurs. En fait, sur cette couche, nous trouvons les utilisateurs eux-mêmes et les commutateurs de couche d'accès. L'objectif principal de cette couche est de connecter physiquement les utilisateurs au réseau. En d'autres termes, il n'y a qu'un câble entre les PC des utilisateurs finaux et les commutateurs de la couche d'accès.

— **La couche de distribution :**La couche de distribution relie les utilisateurs à la couche principale. Il sert d'épine dorsale majeure à tous les utilisateurs d'une zone et permet donc de connecter plusieurs commutateurs d'accès. Dans la plupart des déploiements, les passerelles par défaut de tous les VLAN résident dans la couche de distribution.

— **La couche de base :** Dans l'architecture à trois niveaux, la couche de base est celle qui coordonne tout. Son objectif est simple : connecter toutes les couches de distribution. Dans les grandes entreprises, où vous avez plusieurs commutateurs de distribution, la couche principale est également appelée **Backbone**.

1.2 Travail réalisé

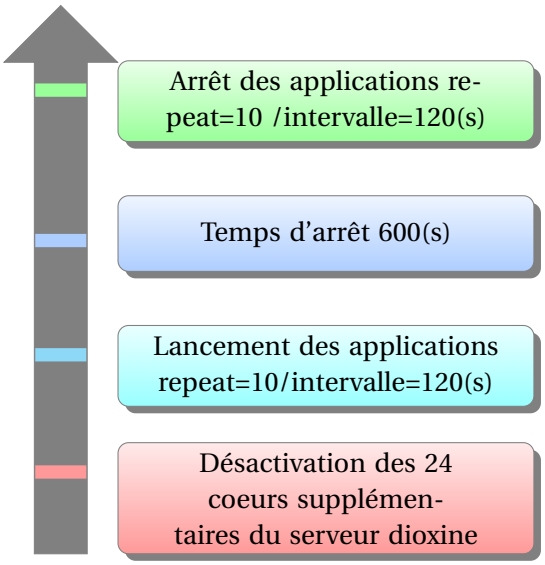
1.2.1 Conditions de travail

Tous d'abord, avant de lancer n'importe quel test sur les serveurs il fallait définir les conditions de travail :

— Nombre de CPU à utiliser sur les serveurs.
— Nombre d'applications à lancer.
— L'intervalle de temps à prendre entre chaque application.
— Le nombre de snap à prendre et l'intervalle entre chaque snap.

Sachant que les deux serveurs ont des microprocesseurs multi-cœur [3] donc pour permettre un équilibre entre les serveurs en terme de la puissance des processeurs, il fallait éteindre les cœurs en plus sur le serveur dioxine et ne lui laisser que les huit premiers ou derniers coeurs.

Ensuite, il fallait définir les intervalles et le nombre de répétitions des applications à lancer, au départ, nous étions parti sur 20 applications avec un intervalle de deux minutes et demie, ce qui nous fait à peu près 1 H 45, ceci est une grande contrainte pour la visualisation des statistiques sur iDRAC. IDRAC nous offre une visualisation des statistiques sur le taux de consommation énergétique plutôt claire sur un intervalle de 1H sur avec l'échelle de 15min. Donc nous sommes passé aux conditions suivantes : 10 applications avec un intervalle de deux minutes entre chaque application ce qui nous fait au total 5O minutes entre le lancement, arrêt et la pause du test.



Finalement, pour générer les log il fallait au moins 50 minutes de captures ce qu'il fait 3000(s), notre choix était de faire 300 captures séparées par des intervalles de 10(s) entre chaque capture.

3. ceux sont des processeur possédant plusieurs cœurs physiques fonctionnant simultanément. Ils se distinguent des architectures plus anciennes (360/91) où un processeur unique commandait plusieurs circuits de calcul simultanés

1.2.2 Serveur dioxine

Serveur rack haute densité 1U extensible avec une conception monosocket efficace pour un faible coût total de possession. Idéal pour la virtualisation dense et le stockage SDS extensible 32C.

Résultat des tests en 3D (temps charge consommation)

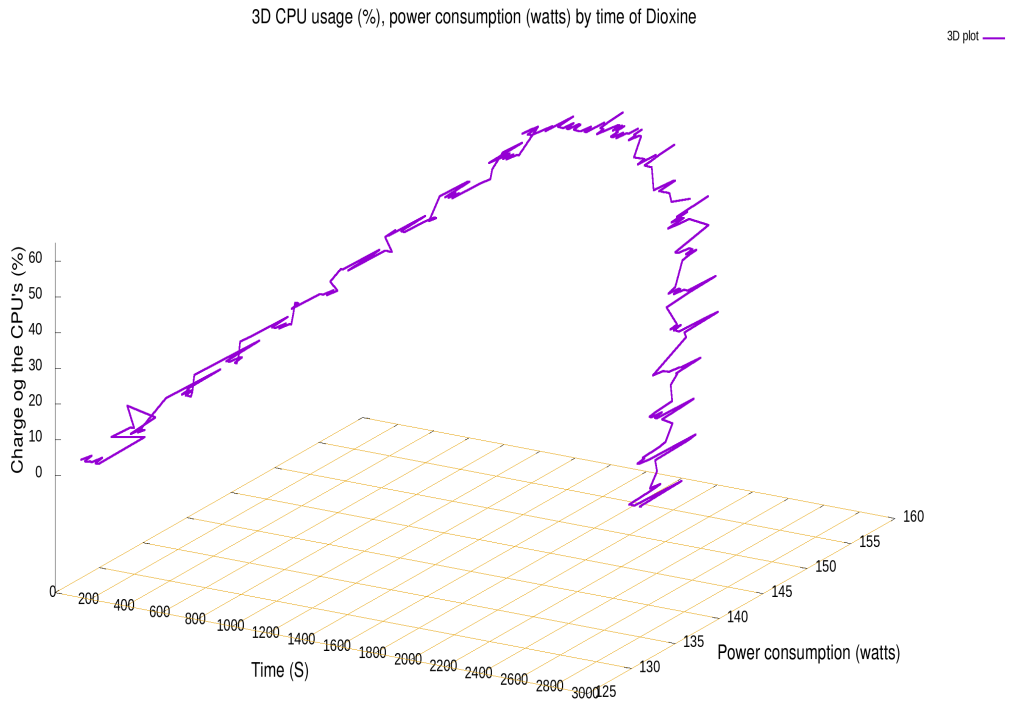


FIGURE 1.5 – Mesure temps, charge et consommation du serveur dioxine

La figure 1.5 représente une courbe 3D qui montre la variation de la charge des CPU et le taux de la consommation énergétique en fonction du temps du serveur dioxine.

On constate tout d'abord que la charge des CPU augmente ou diminue stablement. Et atteint le sommet au milieu du test (c'est le moment où toutes les applications sont lancées).

Ensuite, en ce qui concerne la consommation énergétique on observe que à la fin des tests le décroissement des valeurs n'est pas stable, c'est-à-dire qu'on a des pics de consommation pendant la décroissance.

De plus, on voit bien sur la courbe que le pic est atteint au milieu du test pendant le temps d'arrêt, consommation = 162 watts, charge = 60 %.

Mesure de variation de la charge des CPU en fonction du temps

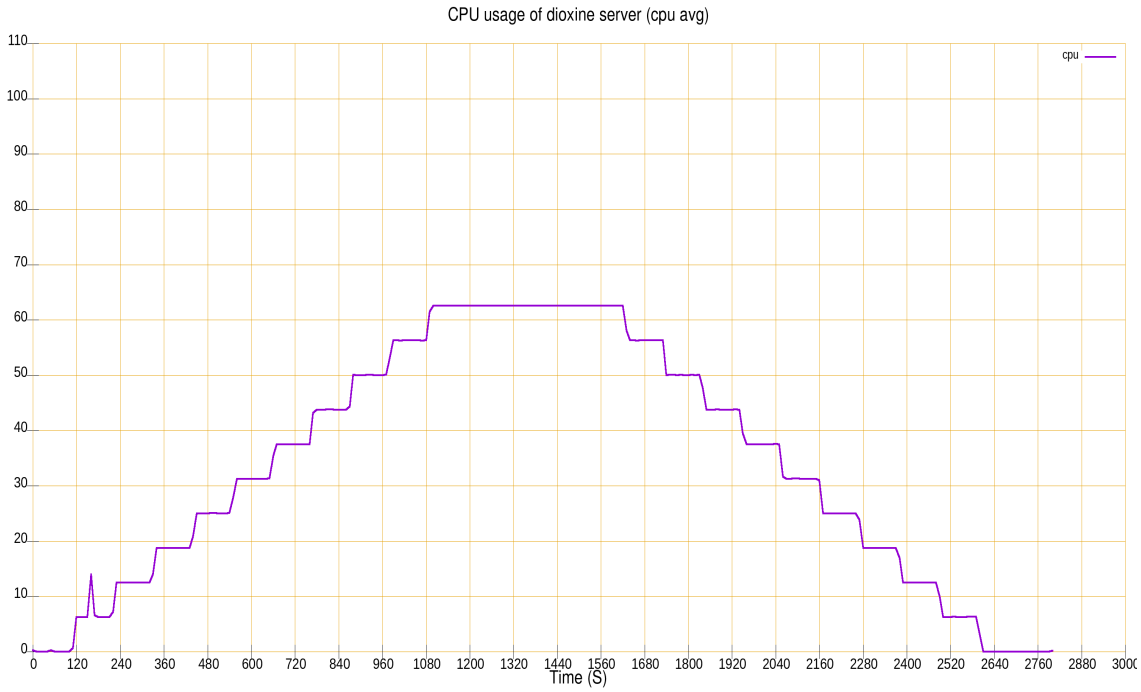


FIGURE 1.6 – La moyenne de la charge des CPU en fonction du temps du serveur dioxine

Mesure de la consommation énergétique en fonction de la charge des CPU

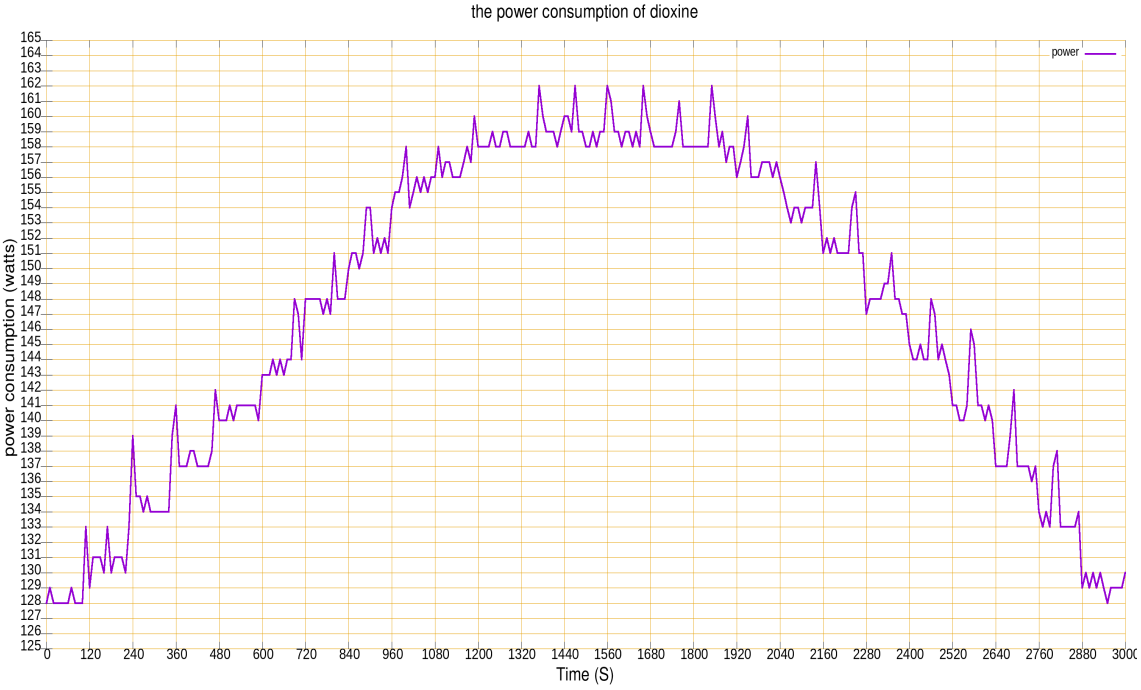


FIGURE 1.7 – La consommation énergétique en fonction du temps du serveur dioxine-IPMI

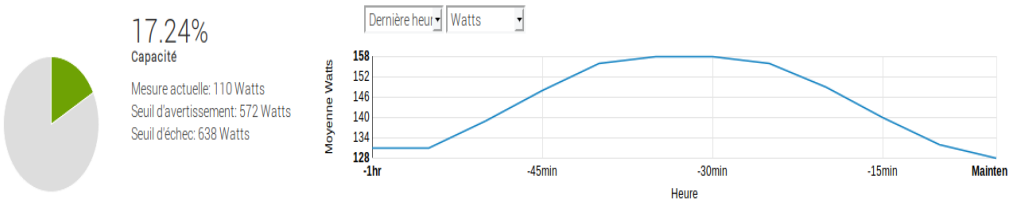


FIGURE 1.8 – La consommation énergétique en fonction du temps du serveur dioxine-iDRAC

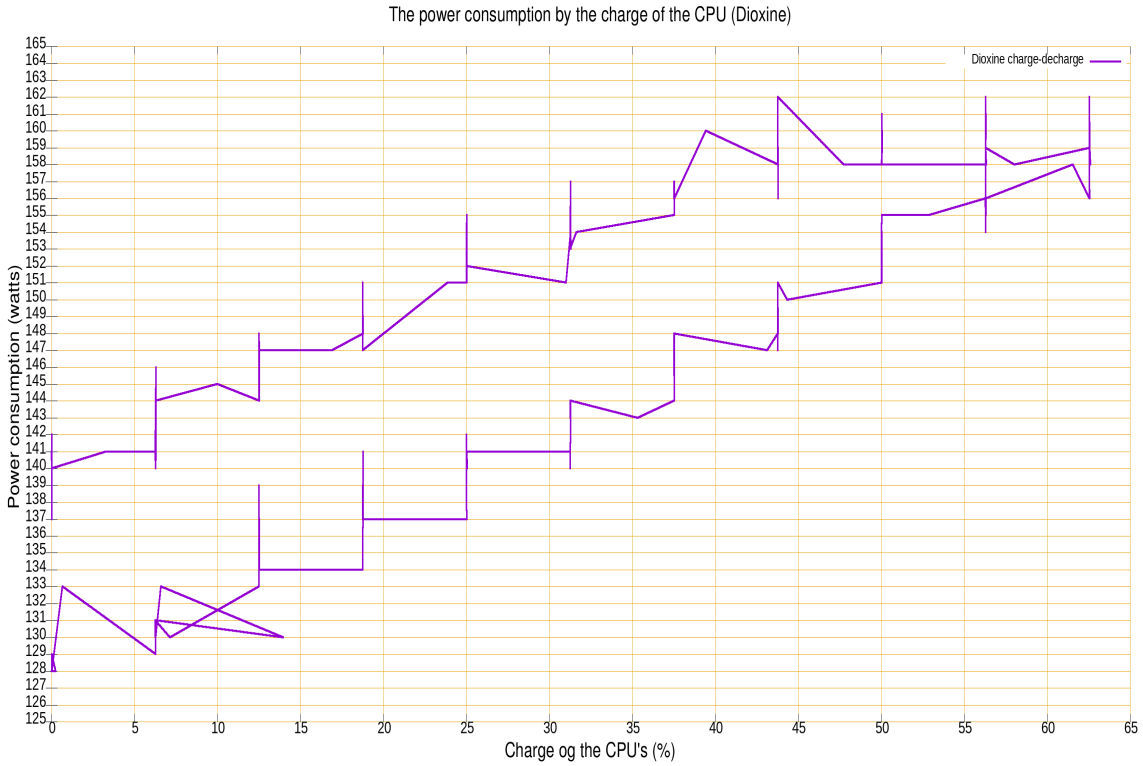


FIGURE 1.9 – La consommation énergétique en fonction de la charge des CPU

La figure 1.9 représente la charge et la décharge du serveur dioxine, la charge augmente plutôt rapidement et elle atteint un pic de consommation égal à 162 (Watts).

1.2.3 Serveur glyphosate

Smart Value PowerEdge R630 Server pour optimiser l'efficacité du datacenter avec un moteur de base de données ou de virtualisation ultradense qui prend en charge jusqu'à 10 disques SSD Flash dans un boîtier 1U 16C.

Résultat des tests en 3D (temps charge consommation)

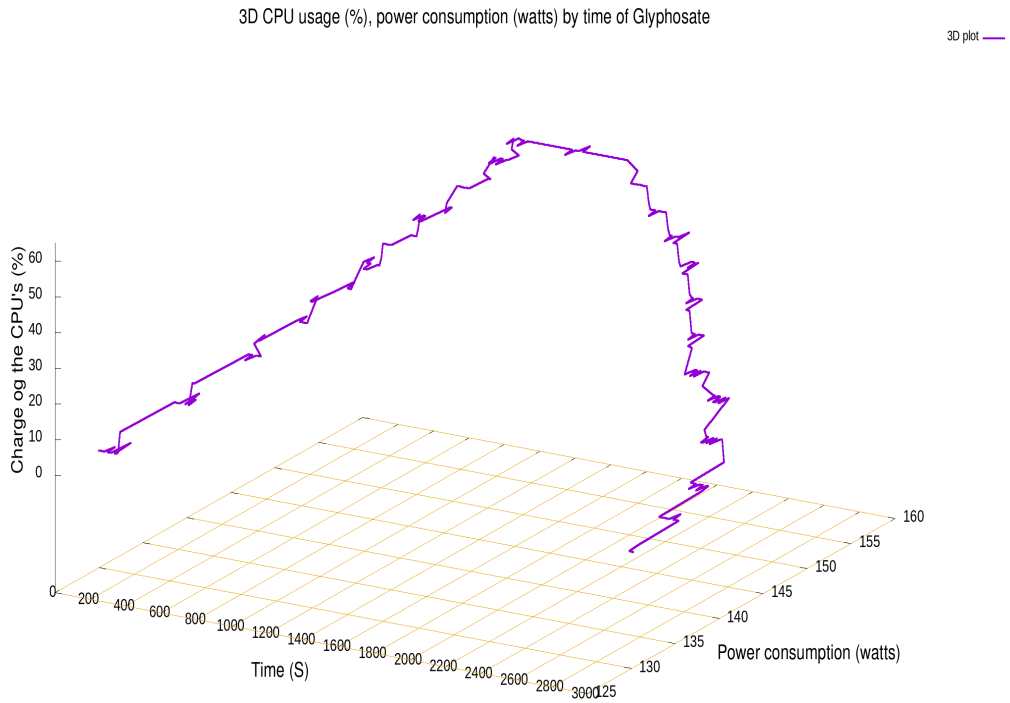


FIGURE 1.10 – Mesure temps, charge et consommation du serveur glyphosate

La figure 1.10 représente une courbe 3D qui montre la variation de la charge des CPU et le taux de la consommation énergétique en fonction du temps du serveur glyphosate.

On constate tout d'abord que la charge des CPU augmente et diminue stablement. Et atteint sommet au milieu du test (c'est le moment ou toutes les applications sont lancées).

Ensuite, en ce qui concerne la consommation énergétique on observe qu'à la fin des tests le décroissement des valeurs n'est pas stable, c'est-à-dire qu'on a des pics de consommation pendant la décroissance.

De plus, on voit bien sur la courbe que le pic est atteint au milieu du test pendant le temps d'arrêt, consommation = 155 Watts, charge = 60 %.

Mesure de variation de la charge des CPU en fonction du temps

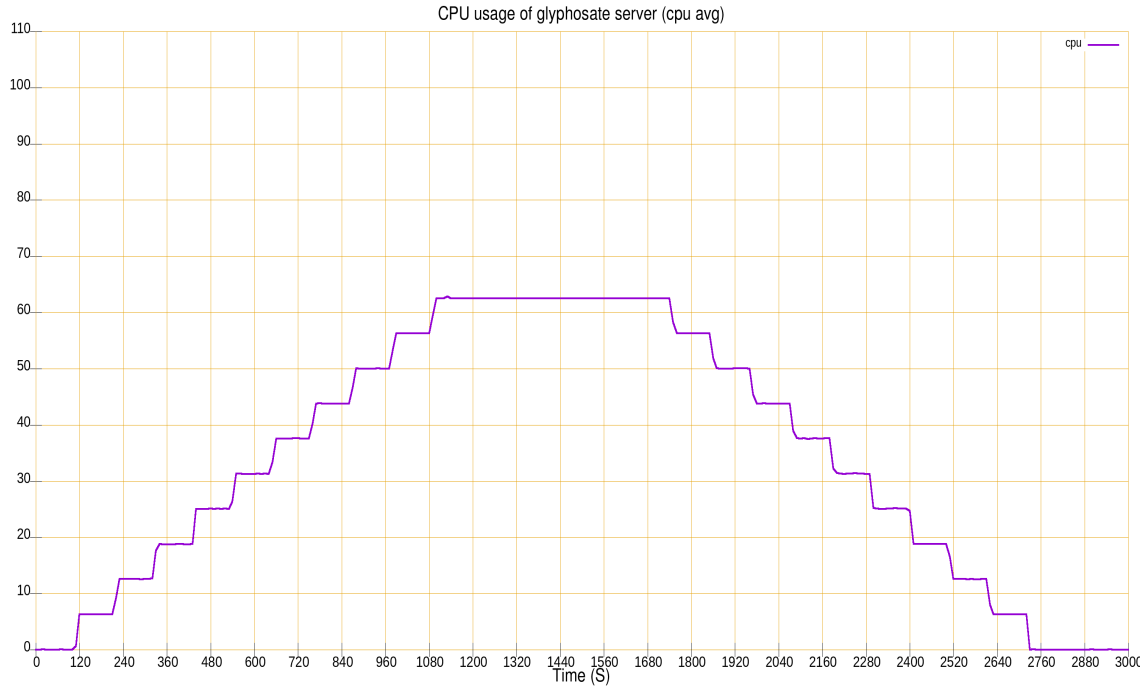


FIGURE 1.11 – La moyenne de la charge des CPU en fonction du temps du serveur glyphosate

Mesure de la consommation énergétique en fonction de la charge des CPU

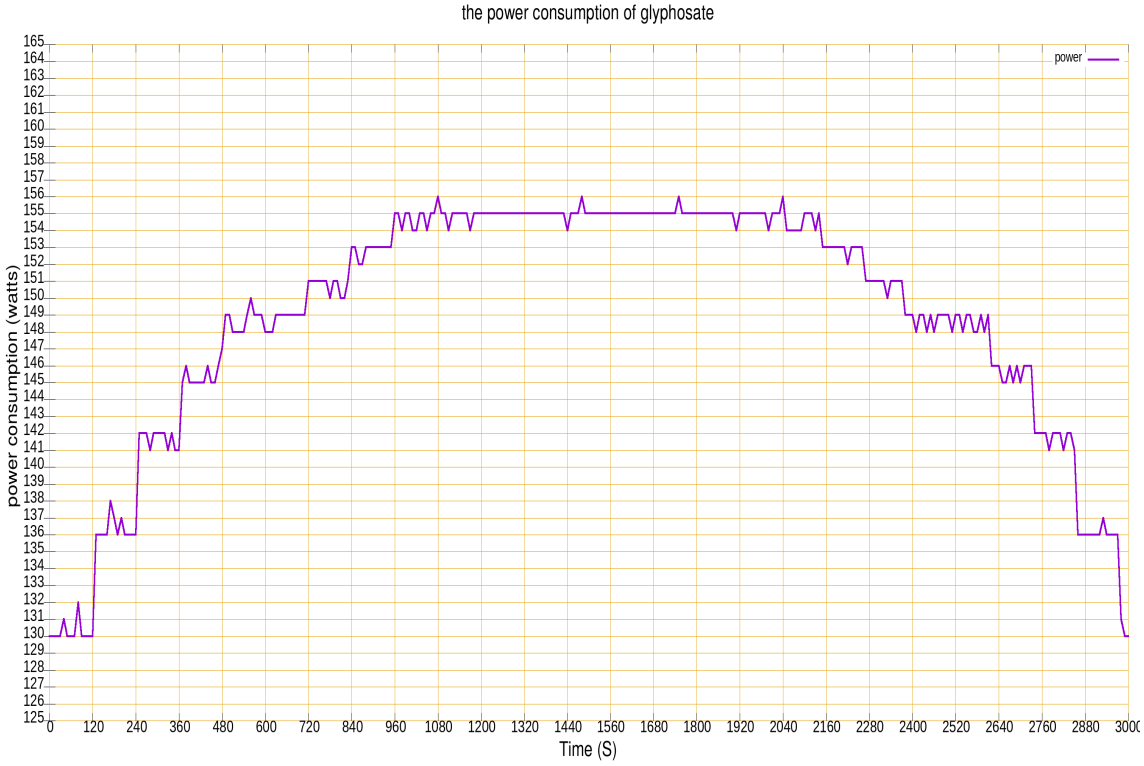


FIGURE 1.12 – La consommation énergétique en fonction du temps du serveur glyphosate-IMPI

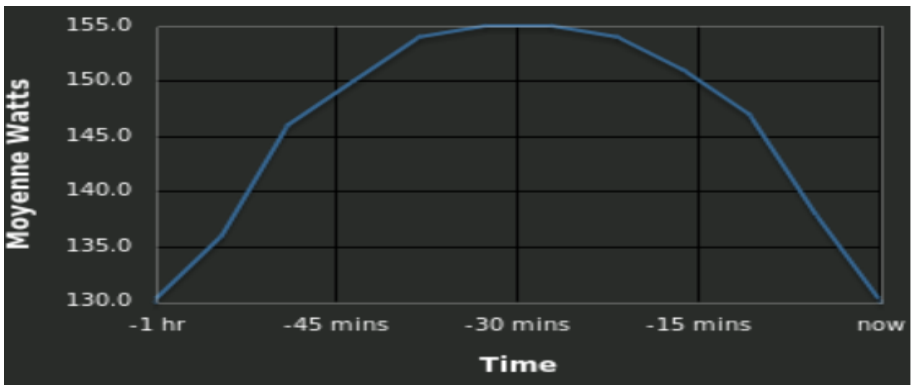


FIGURE 1.13 – La consommation énergétique en fonction du temps du serveur glyphosate-iDRAC

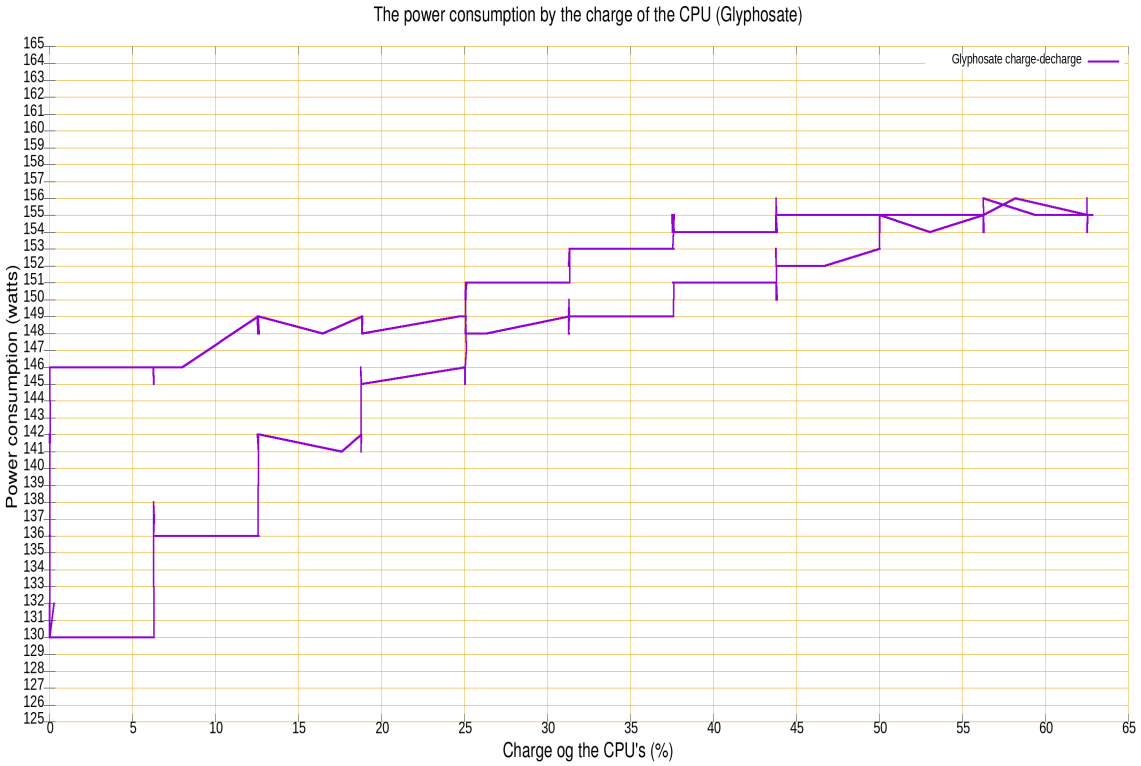


FIGURE 1.14 – La consommation énergétique en fonction de la charge des CPU

La figure 1.14 représente la charge et la décharge du serveur glyphosate, l'augmentation est moins brusque que celle du serveur dioxine (voir figure 1.9) et elle est plutôt stable. La consommation énergétique atteint son maximum 155 (watts) puis se décharge.

Chapitre 2

La partie technique du stage

2.1 Travail réalisé

Chapeau Dans cette section, je présente la partie technique et les apport sur le projet de mon stage, cela en donnant des explications sur les scripts qui vont avec ces derniers.

2.1.1 Mesure de variation de la charge des CPU en fonction du temps

Comme les serveurs sont des serveurs ubuntu, la mesure de variation de la charge des CPU en fonction du temps se base sur le système de fichiers proc [1], ceci en utilisant le fichiers de statistiques **/proc/stat**.

Diverses informations sur l'activité du noyau sont disponibles dans le fichier /proc/stat. Tous les nombres rapportés dans ce fichier sont des chargé depuis le premier démarrage du système.

Pour un aperçu rapide, il suffit de lire le fichier :

```
> cat /proc/stat
cpu  2255 34 2290 22625563 6290 127 456
cpu0 1132 34 1441 11311718 3675 127 438
cpu1 1123 0 849 11313845 2614 0 18
intr 114930548 113199788 3 0 5 263 0 4 [... lots more numbers ...]
ctxt 1990473
btime 1062191376
processes 2915
procs_running 1
procs_blocked 0
```

FIGURE 2.1 – Le contenu du fichier /proc/stat

Ces chiffres identifient le temps passé par la CPU à effectuer différents types de travail. Les unités de temps sont exprimées en USER_HZ ou Jiffies (généralement des centièmes de seconde).

La signification des colonnes est la suivante, de gauche à droite :

1. user : processus normaux s'exécutant en mode utilisateur
2. nice : processus en cours d'exécution en mode utilisateur
3. system : processus s'exécutant en mode noyau
4. idle : les pouces qui tournent
5. iowait : en attente de la fin des E / S

1. Le système de fichiers proc est un pseudo-système de fichiers qui fournit une interface avec les structures de données du noyau. Il est généralement monté à /proc.

6. irq : le service des interruptions
7. softirq : entretien des logiciels

Le script [3.2] pour une temps (t) lit deux fois le contenu du fichier **/proc/stat** pour générer deux états (état courant, état précédant) puis effectue le calcul suivant :

```
    PrevNonIdle=$((prevuser + prevnice + prevsystem + previrq + prevsoftirq + prevsteal))
    NonIdle=$((user + nice + system + irq + softirq + steal))

    PrevTotal=$((PrevIdle + PrevNonIdle))
    Total=$((Idle + NonIdle))

    totald=$((Total - PrevTotal))
    idled=$((Idle - PrevIdle))

    if [ $totald -eq $zero ]; then
        CPU_Percentage="0"
    else
        CPU_Percentage=$(awk "BEGIN {print ($totald - $idled)/$totald*100}")
    fi

```

Listing 2.1 – Calcul de la charge des CPU

Les résultats sont stocké dans deux fichiers, un fichier log avec plus de texte et un fichier dat avec moins de texte pour afficher les courbes en utilisant Gnuplot [2].

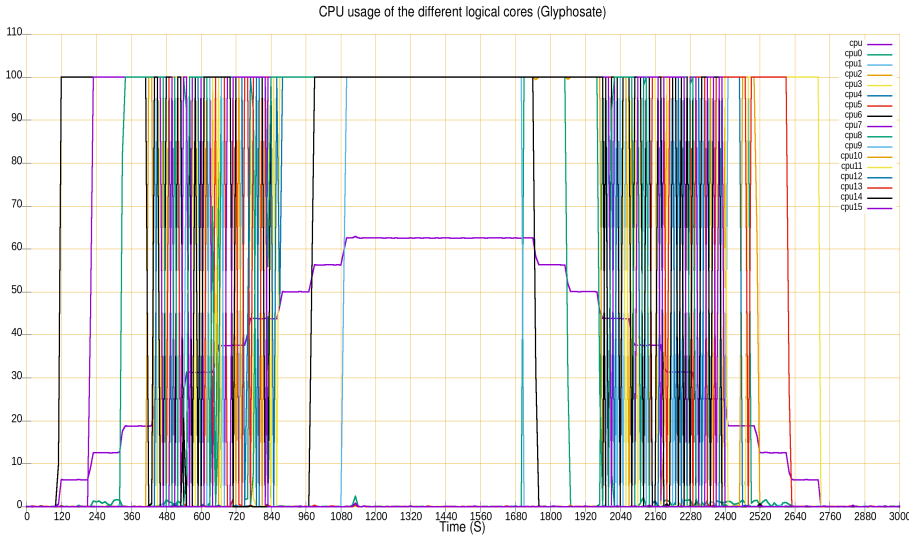


FIGURE 2.2 – CPU usage des coeurs logiques du serveur Glyphosate

Mais le script récupère que les taux d'utilisations des coeurs logiques. En utilisant ces fichiers on peut appliquer divers traitement, dans un premier lieu la récupération des coeurs physiques est primordial pour bien comparer les comportements des deux serveurs. Le programme C [3.1] s'en occupe de ça.

2. Gnuplot est une puissante interface pour la représentation graphique de données provenant d'un fichier texte. Il peut soit afficher l'image, soit enregistrer une image dans divers formats, y compris LaTeX.

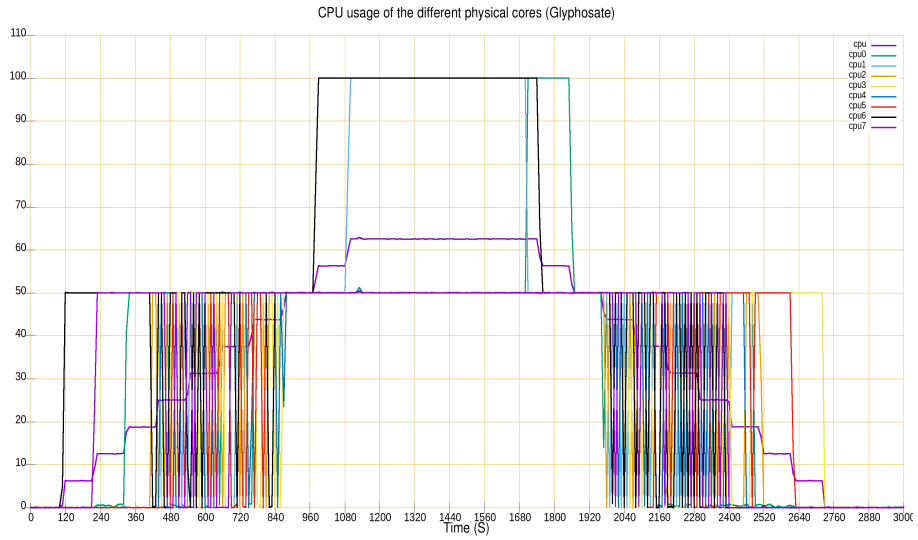


FIGURE 2.3 – CPU usage des coeurs physiques du serveur Glyphosate

Ceci des fois n'est pas claire pour bien observer les croissances et les pics d'utilisation. Pour cela j'ai mis en place un script C qui permet de séparer les log d'un coeur précis.

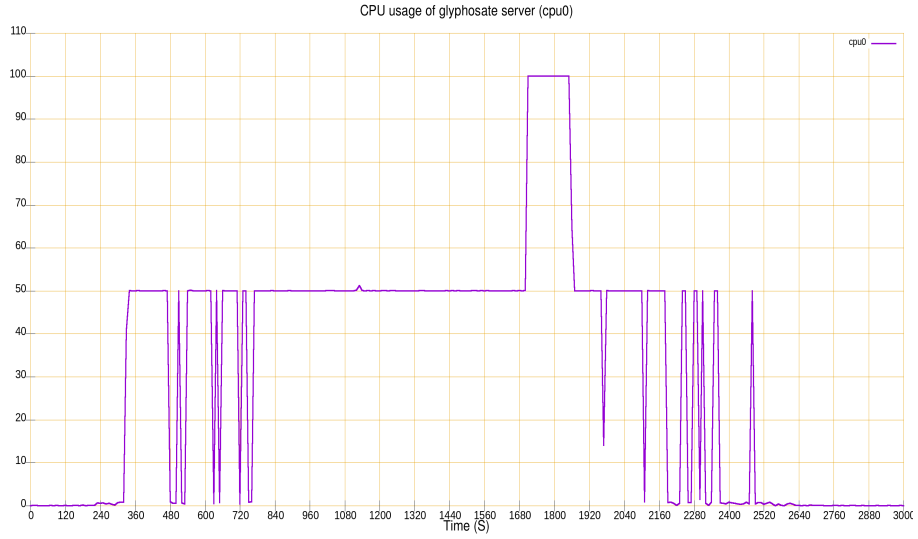


FIGURE 2.4 – CPU usage du coeur 0 du serveur Glyphosate

2.1.2 Mesure de la consommation énergétique

Pour Mesurer la consommation énergétique des serveurs et laisser des traces (fichiers log) j'ai utilisé IPMI [3]. Avec IPMI on peut :

1. Récupérer la consommation électrique d'un serveur via une ligne de commande
2. Accéder via un flux réseau, à l'affichage de la console du serveur
3. Prise en main a distance par la console

3. IPMI est une interface de gestion de matériel standardisée (Intelligent Platform Management Interface)fournie principalement sur les serveurs (Dell, IBM, HP, Intel, NEC, Supermicro, etc), indépendante du système d'exploitation, et destinée à contrôler certains composants hardware.

Le script [3.3] prend en paramètre deux argument, **numberOfsnap** représente le nombre de captures a effectuer et **sleepDurationSeconds** qui représente l'intervalle de temps entre deux captures, pour obtenir les résultats présenté sur la figure [3.1].

On à aussi utilisé iDrac pour récupérer les courbes de consommations énergétique des serveurs comme présenté dans les figures [1.13, 1.8], puis avec ça on peut faire des conclusions sur les deux résultats obtenus et en tenir une conclusion détaillé dans la section 3.1.1.

2.1.3 Manager des applications

Pour effectuer nos comparaison et définir un plan de travail similaire aux deux serveurs, il fallait lancer des applications pour charger les différents coeurs des CPU des deux serveurs. Pour cela j'ai mis en plac un script qui gère les applications, avec ce script on peut :

— Lancer un nombre d'applications avec un intervalle de temps entre deux applications.
— stopper un nombre d'applications avec un intervalle de temps entre deux applications.

Le script demande aux utilisateurs de rentrer : **[app] [mode] [apply] [repeat] [interval]**

1. app : le nom de l'application a traité. [*.c | script]
2. mode : le mode de l'application, soit un exécutable ou bien un script. [e|s]
3. apply : le traitement a "effectuer, soit lancer ou stopper. [run|kill]
4. repeat : le nombre d'applications a lancé. [Integer]
5. interval : l'intervalle de temps entre deux applications. [Integer]

Les différents applications disponibles sur le script sont :

1. dd : "dd if=/dev/zero of=/dev/null" :
   — /dev/zero fournit un flux sans fin de zéro octet lors de la lecture. Cette fonction est fournie par le noyau et ne nécessite pas d'allocation de mémoire. Toutes les écritures dans /dev/null sont supprimées silencieusement.
   — Données en double (dd) du fichier d'entrée (if) de /dev/zero (fourniture virtuelle illimitée de 0) dans le fichier de sortie (de) de /dev/null (trou noir virtuel) à l'aide de blocs de taille 500M (bs = taille de bloc) et Répétez ceci (compter) une seule fois (1)
2. rand : "while true; do echo $((13**99)) 1> /dev/urandom 2>&1; done" :
   — 100% de charge sur une machine Linux.
   — Via un moment (un ou plusieurs CPU).
3. one_full : "while true; do true; done" :
   — Via une utilisation de temps de 50% du processeur
4. cat : "cat /dev/urandom > /dev/null" :
   — Via une commande cat.
   — Bien que vos caractères ASCII imprimables normaux soient échangés dans un terminal, de nombreux caractères non imprimables sont également utilisés pour que le système communique avec le terminal. Par exemple, si un programme envoie le caractère 0x07 ("caractère ASCII de Bell"), votre terminal doit émettre un bip.

2.1.4 Désactiver les CPU

Sachant que les deux serveurs ont des microprocesseurs multi-cœur donc pour permettre un équilibre entre les serveurs en terme de la puissance des processeurs, il fallait éteindre les cœurs en plus sur le serveur dioxine et ne lui laisser que les huit premiers ou derniers coeurs.

Pour gérer cela j'ai mis en place un script qui peut activer, désactiver et afficher les CPU qui sont activer la même chose que pour les script présenté dans la section [2.1.3], le script demande aux utilisateurs de rentrer : [on|off] [begin|end] [Integer]

1. (on|off) : veut dire activer ou désactiver les CPU.
2. (begin|end) : (from begin, from end), activer ou désactiver les CPU depuis le début ou depuis la fin.
3. (Integer) : le nombre de CPU a désactivé.

exemple d'utilisation :

```
$./cpu_deactivate.sh
[on|off] [begin|end] [Integer]
Enter your choice: cpu
processor : 0
processor : 1
processor : 2
processor : 3
processor : 4
processor : 5
processor : 6
processor : 7
Enter your choice: off end 3
Enter your choice: cpu
processor : 0
processor : 1
processor : 2
processor : 3
processor : 4

```

Listing 2.2 – exemple d'utilisation du script qui désactiver les CPU

Enfin, pour combiner tout cela et pour respecter les conditions de travail présentés dans la section [1.2.1] j'ai mis en place le script [3.4], qui vient pour lancer les deux script (mesure de la consommation énergétique en fonction du temps, mesure de la charge des CPU en fonction du temps) avec 3000 capture et un intervalle de 10 (S) entre deux captures.

Chapitre 3

Conclusion

3.1 Conclusion technique

Chapeau Dans cette section, je résume la réalisation du projet et je présente également une comparaison entre les différents outils et matériels.

3.1.1 Comparaison entre IPMI et iDRAC

— IPMI est une interface de gestion de matériel standardisée alors que iDRAC est un contrôleur.
— iDRAC est un contrôleur (integrated Dell remote access controller), il aide a déployer, mettre à jour, surveiller et entretenir les serveurs Dell powerEdge avec ou sans agent logiciel de gestion des systèmes.
— Avec IPMI on a pu générer des fichier log et les utiliser pour appliquer différents traitements comme afficher la courbe 3D et la consommation énergétique en fonction de la charge des CPU.
— iDRAC n'offre pas une trace (un fichier log) il nous affiche que les courbes on ne garde pas des traces.
— l'échelle avec iDRAC n'est pas claire ce qui implique que les pics ne sont pas visibles.
— L'avantage principale de IPMI est le stockage des données pour les utiliser. Par ailleurs on a une visualisation des changements des valeurs plus claires que celle de iDRAC.

3.1.2 Comparaison entre les serveurs

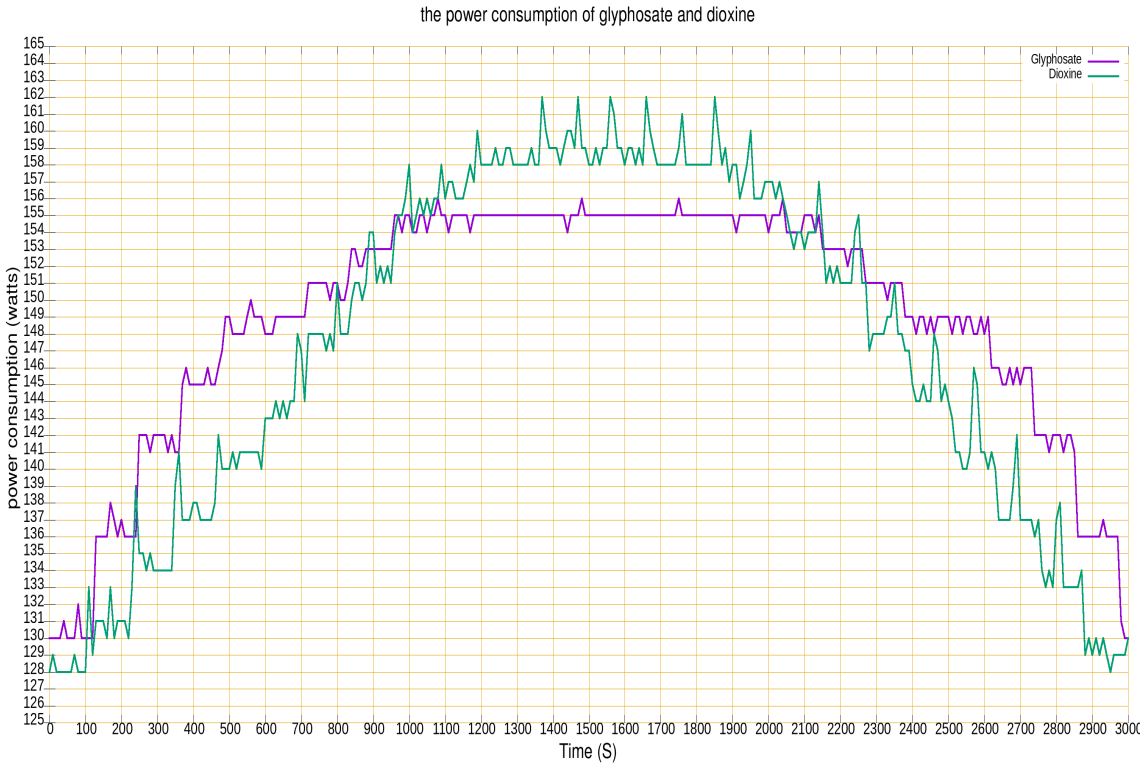


FIGURE 3.1 – Comparaison entre la consommation énergétiques entre les deux serveurs

On observe que la différence en termes de la consommation énergétique entre les serveurs est importants. En effet, il y a un écart important entre les deux serveurs aux milieux des tests. Le serveur **Dioxine** a atteint 162 watts et **Glyphosate** a atteint 155 watts. En ce qui concerne la charge des CPU comme le montre les figures [1.6, 1.11] les moyennes sont à peu-prêt identique avec une légère augmentation sur le serveur **Dioxine**.

De plus, la charge et la décharge du serveur dioxine s'effectuent brusquement le contraire du serveur glyphosate qui prend son temps à se charger et à se décharger.

Enfin, le serveur glyphosate est plutôt stable quand il atteint le temps de pause c'est-à-dire il ne réagit pas avec des pics il reste stable sur la même valeur 155 (watts), au contraire le serveur dioxine réagis avec des pics pendant le temps de pause.

La figure 3.2 représente la variation de la consommation énergétique en fonction de la charge des CPU des serveurs, cette dernière vient nous confirmer les conclusions déduites précédemment. La différence en termes de la consommation énergétique en fonction de la charge des CPU des deux serveurs pendant les tests est claire et en même temps importante, on voit bien que le serveur glyphosate commence avec une consommation plus importante que le serveur dioxine, mais au fil du temps la consommation énergétique du serveur dioxine prend l'élan et augment plus rapidement que la consommation de l'autre serveur, cependant la consommation énergétique du serveur glyphosate grimpe avec une stabilité remarquable.

De plus, en ce qui concerne la décharge, le serveur glyphosate prend son temps à se stabiliser au début de la décharge puis aux milieu de décharge la constante, mais à la fin elle est brusque,

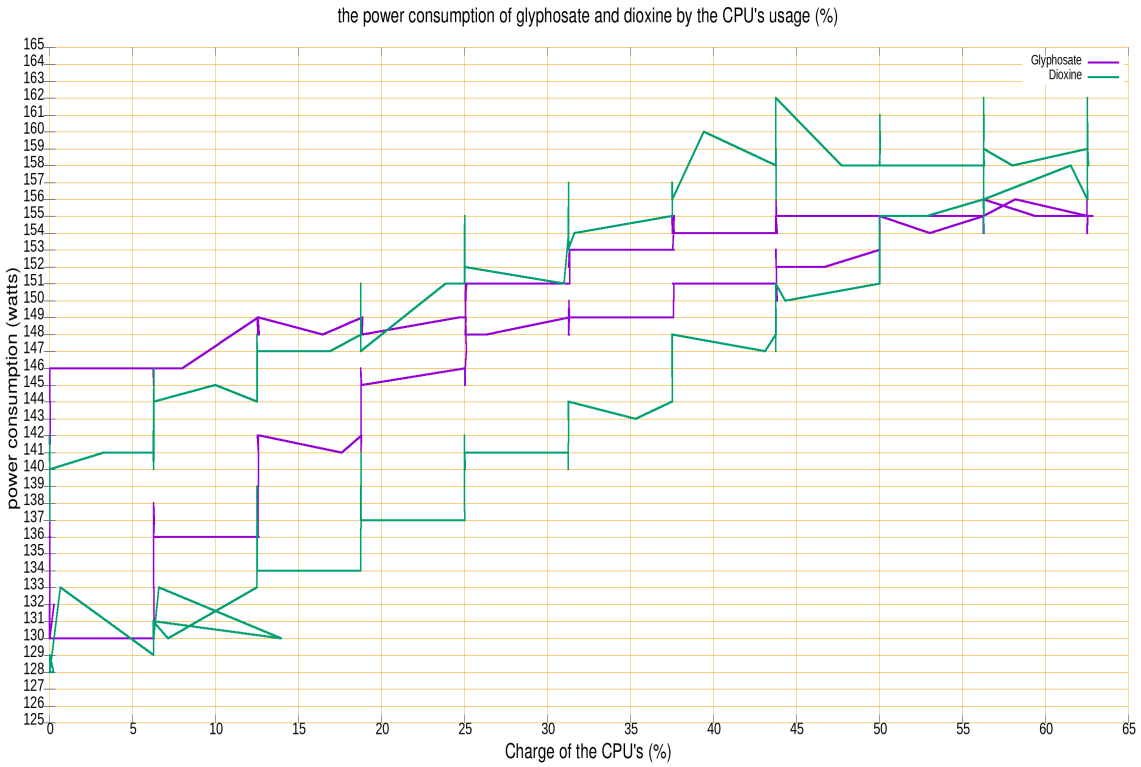


FIGURE 3.2 – Comparaison entre la consommation énergétique en fonctio de la charge des CPU entre les deux serveurs

alors que l'autre serveur la décharge est brusque.

Après cet ensemble de réflexions sur la Consommation Énergétique et la Charge des CPU en fonction du temps, il est assez difficile d'en soutirer une conclusion. C'est pourquoi j'ai décidé de proposer un avis global au lieu de proposer une véritable conclusion au proprement parlé. À mon avis le serveur glyphosate est plus intéressant en terme de consommation énergétique moyenne, mais en ce qui concerne la rapidité de la décharge le serveur dioxine est plus performant.

3.2 Conclusion

Chapeau Dans cette section, je met en oeuvre le point final du stage, cella en détaillant la réalisation du projet et en récapitulant la gestion du temps et les tâches effectuées.

A titre de conclusion, il semble intéressant de mettre en évidence les questions actuelles qui se posent sur l'avenir de l'industrie des centres de données, de savoir si les data centers sont des gouffres énergétiques. Au centre de cette question se trouve naturellement le problème de la consommation énergétique et la surcharge des CPU. En effet, mon stage a été très bénéfique à cet égard : la consommation énergie tique des centres de données, savoir la charge des CPU et en laisser des log. en revanche j'ai pas pu réaliser quelques points que je voulais élaborer comme la Récupérer la chaleur des data centers.

La prochaine étape c'est d'étudier les modèle de trafic de navigation Web pour la mesure de la simulation, ainsi d'étudier les modelés de trafic des utilisateurs de serveurs WEB. La qualité de service (QoS) qui fait référence à la capacité d'un réseau à offrir un service avec une certaine qualité. La qualité d'un service peut être liée à un certain nombre de paramètres différents, aussi d'étudier le comportement des utilisateurs et modèles de navigation Web

√ Les résultats que j'ai eu et que j'ai présenté ne sont pas que des simples courbes / chiffres ils ils peuvent étres très utils. Les fichiers de log si on en a assez d'information sur une journées de travail normal on pourra prédire le comportement des serveurs, cela peut êtres utilisé pour la sécurité et surtout pour prédire une DDOS attaque, Une Dos ou DDos consiste à **épuiser les ressources** à disposition d'un serveur, le plus souvent celles liées au réseau. Cependant, il est également possible de saturer le disque ou la RAM ou encore les processeurs des machines visées.

Donc en utilisant les fichiers log et en déduisant un comportement des serveurs on pourra détecter une attaque DDOS l'annoncer.

3.3 Annexe

Chapeau Dans cette section, je mets en valeur les scripts et les schémas indispensables utilisés sur mon stage.

```c
#include<stdio.h>
#include<stdlib.h>
#include <unistd.h>

#define FUNC_ERROR -1

/**
*This fonction takes a table of logical cpu's data ana calcul the physical data of them.
*@param nopc is the number of physical cpu's
*@param tab_data_logical the logical cpu data
*@return a table how contains the physical data of the cpu's
*/

float* table_convert(int nopc, float* tab_data_logical){
  float* tab_data_physical = malloc(sizeof(float) * (nopc - 1));
  if(tab_data_physical == NULL){
    fprintf(stderr, "table_convert: malloc error!\n");
    return NULL;
  }

  for(int i = 0; i < nopc; i++){
    tab_data_physical[i] = (tab_data_logical[i] + tab_data_logical[i + nopc]) / 2;
  }
  return tab_data_physical;
}

/**
*This function is used to convert the data of logical cpu's to an physical data cpu's
*@param nopc is the number of physical cpu's
*@return it's returns 0 if evrithing is done, FUNC_ERROR if ther's a problem
*data_logical.dat is the name of the file how contains the data of the logical cpu's
*/

int convert_to_physical(int nopc){
  FILE* file_data_logical;
  FILE* file_data_physical;
  float time, all;
  char * line = NULL;
    size_t len = 0;
    ssize_t read;
    float* tab_data_physical;

  file_data_logical = fopen ("../plot/data_logical.dat", "r+");
  file_data_physical = fopen("../plot/data_physical.dat", "w+");

  if(file_data_logical == NULL || file_data_logical == NULL){
    fprintf(stderr, "convert_to_physical: open file data error!\n");
    return FUNC_ERROR;
  }

  if(nopc < 2){
    fprintf(stderr, "convert_to_physical: invalid argument!\n");
    return FUNC_ERROR;
  }

  float* tab_data = malloc(sizeof(float) * ((nopc * 2) - 1));
```

```
57    if(tab_data == NULL){
58      fprintf(stderr, "convert_to_physical: malloc error!\n");
59      return FUNC_ERROR;
60    }
61
62    fprintf(file_data_physical, "t cpu ");
63    for(int i = 0; i < nopc; i++){
64      fprintf(file_data_physical, "cpu%d ", i);
65    }
66
67    while((read = getline(&line, &len, file_data_logical)) != -1){
68      fscanf(file_data_logical, "\n%f %f ", &time, &all);
69      fprintf(file_data_physical, "\n%f %f ", time, all);
70
71      for(int i = 0; i < nopc * 2; i++){
72        if(i < (nopc * 2) -1)
73            fscanf(file_data_logical, "%f ", tab_data+i);
74          else
75            fscanf(file_data_logical, "%f", tab_data+i);
76        }
77        tab_data_physical = table_convert(nopc, tab_data);
78        for (int i = 0; i < nopc; i++){
79          fprintf(file_data_physical, "%f ", tab_data_physical[i]);
80        }
81    }
82
83    if (line)
84      free(line);
85    fclose(file_data_logical);
86    fclose(file_data_physical);
87    if (tab_data_physical)
88      free(tab_data_physical);
89    free(tab_data);
90    return 0;
91  }
92
93
94  int main(int argc, char** argv){
95
96    if(argc < 2){
97      fprintf(stderr, "main: invalid argument!\n");
98      printf("usage: %s [number of physical cpu]\n", argv[0]);
99      return FUNC_ERROR;
100   }
101
102   if(convert_to_physical(atoi(argv[1])) < 0){
103     fprintf(stderr, "main: convert_to_physical!\n");
104     return FUNC_ERROR;
105   }
106
107   return 0;
108 }
109
```

Listing 3.1 – Script pour le calcul des charges des coeurs physiques

```
#!/bin/bash

sleepDurationSeconds=$1
numberOfsnap=$2

i=0
x=0
zero=0

#command to get the number of core
CORE_NUMBER=$(grep -c ^processor /proc/cpuinfo)
#put the interval and the number of cors into a string
INTERVAL="interval=$1 -- number of cors=$CORE_NUMBER"
echo $INTERVAL > ../log/cpu_usage.log

while [ $i -lt $numberOfsnap ]
do
    previousStats=$(cat /proc/stat)

    sleep $sleepDurationSeconds

    currentDate=$(date "+%Y-%m-%d %H:%M:%S")

    CPU_USAGE="----------$currentDate----------\n BEGIN\n"

    currentStats=$(cat /proc/stat)

    cpus=$(echo "$currentStats" | grep -P 'cpu' | awk -F " " '{print $1}')

    if [ $i -eq $zero ]; then
        tmp="t $cpus"
        echo  $tmp > ../plot/data_logical.dat
    fi

    data_plot="$x"
    for cpu in $cpus
    do
        currentLine=$(echo "$currentStats" | grep "$cpu ")
        user=$(echo "$currentLine" | awk -F " " '{print $2}')
        nice=$(echo "$currentLine" | awk -F " " '{print $3}')
        system=$(echo "$currentLine" | awk -F " " '{print $4}')
        idle=$(echo "$currentLine" | awk -F " " '{print $5}')
        iowait=$(echo "$currentLine" | awk -F " " '{print $6}')
        irq=$(echo "$currentLine" | awk -F " " '{print $7}')
        softirq=$(echo "$currentLine" | awk -F " " '{print $8}')
        steal=$(echo "$currentLine" | awk -F " " '{print $9}')
        guest=$(echo "$currentLine" | awk -F " " '{print $10}')
        guest_nice=$(echo "$currentLine" | awk -F " " '{print $11}')

        previousLine=$(echo "$previousStats" | grep "$cpu ")
        prevuser=$(echo "$previousLine" | awk -F " " '{print $2}')
        prevnice=$(echo "$previousLine" | awk -F " " '{print $3}')
        prevsystem=$(echo "$previousLine" | awk -F " " '{print $4}')
        previdle=$(echo "$previousLine" | awk -F " " '{print $5}')
        previowait=$(echo "$previousLine" | awk -F " " '{print $6}')
        previrq=$(echo "$previousLine" | awk -F " " '{print $7}')
        prevsoftirq=$(echo "$previousLine" | awk -F " " '{print $8}')
        prevsteal=$(echo "$previousLine" | awk -F " " '{print $9}')
        prevguest=$(echo "$previousLine" | awk -F " " '{print $10}')
        prevguest_nice=$(echo "$previousLine" | awk -F " " '{print $11}')
```

```
61
62          PrevIdle=$((previdle + previowait))
63          Idle=$((idle + iowait))
64
65          PrevNonIdle=$((prevuser + prevnice + prevsystem + previrq + prevsoftirq + prevsteal
    ))
66          NonIdle=$((user + nice + system + irq + softirq + steal))
67
68          PrevTotal=$((PrevIdle + PrevNonIdle))
69          Total=$((Idle + NonIdle))
70
71          totald=$((Total - PrevTotal))
72          idled=$((Idle - PrevIdle))
73
74          if [ $totald -eq $zero ]; then
75              CPU_Percentage="0"
76          else
77              CPU_Percentage=$(awk "BEGIN {print ($totald - $idled)/$totald*100}")
78          fi
79
80          CPU_USAGE="$CPU_USAGE\t\t$cpu $CPU_Percentage%\n"
81          data_plot="$data_plot $CPU_Percentage"
82      done
83
84      CPU_USAGE="$CPU_USAGE DONE"
85
86      echo $CPU_USAGE >> ../log/cpu_usage.log
87      echo $data_plot >> ../plot/data_logical.dat
88
89      i=$(($i + 1))
90      x=`echo "$x + $sleepDurationSeconds" | bc -l`
91  done
92
```

Listing 3.2 – Script Mesure de la charge des CPU

```
1  #!/bin/bash
2
3  sleepDurationSeconds=$1
4  numberOfsnap=$2
5
6  i=0
7  x=0
8
9  currentDate=$(date "+%Y-%m-%d %H:%M:%S")
10 date="------------------$currentDate------------------"
11 echo $date > ../log/power_consumption.log
12
13 while [ $i -lt $numberOfsnap ]
14 do
15     sleep $sleepDurationSeconds
16
17     ipmitool dcmi power reading | sed -e "s/  */ /g" >> ../log/power_consumption.log
18
19     if [ $i -eq 0 ]; then
20         tmp="t power"
21         echo  $tmp > ../plot/power_consumption.dat
22     fi
23
24     power_2="$x"
25     watts=$(ipmitool dcmi power reading | sed -e "s/  */ /g" | grep 'Instantaneous power
```

```
    reading' | cut -d' ' -f5)

    power_2="$power_2 $watts"

    echo $power_2 >> ../plot/power_consumption.dat

    i=$(($i + 1))
    x=`echo "$x + $sleepDurationSeconds" | bc -l`
done
```

Listing 3.3 – Script Mesure de la consommation énergétique

```
#!/bin/bash

interval=10
repeat=3000

./cpu_usage.sh $interval $repeat &
./power_consumption.sh $interval $repeat &
./app_manager.sh

cut -d' ' -f2 ../plot/data_logical.dat | paste -d' ' ../plot/power_consumption.dat - > ../plot/xyz.dat
awk '{print $3,$2}' ../plot/xyz.dat > ../plot/cpu_usage_power.data
```

Listing 3.4 – Script pour lancer les tests

```
#!/bin/bash

quit=0
help=1
number_core_physical=$1

function disable_enable {
  com="/sys/devices/system/cpu/cpu$2/online"
  echo $1 | sudo tee $com
}

function on_cpu {
  case $1 in
    begin )
      i=$2
      while [ $i -le $(($number_core_physical - 1)) ]; do
        core_physical=$(($i + $number_core_physical))
        disable_enable 1 $i
        disable_enable 1 $core_physical
        i=$(($i + 1))
      done
    ;;
    end )
      i=$(($(($number_core_physical - 1)) - $2))
      while [ $i -ge 0 ]; do
        core_physical=$(($i + $number_core_physical))
        disable_enable 1 $i
        disable_enable 1 $core_physical
        i=$(($i - 1))
      done
```

```
31      ;;
32      * ) help=1 ;;
33    esac
34  }
35
36  function off_cpu {
37    case $1 in
38      begin )
39        i=$2
40        while [ $i -le $(($number_core_physical - 1)) ]; do
41          core_physical=$(($i + $number_core_physical))
42          disable_enable 0 $i
43          disable_enable 0 $core_physical
44          i=$(($i + 1))
45        done
46      ;;
47      end )
48        i=$(($(($number_core_physical - 1)) - $2))
49        while [ $i -ge 0 ]; do
50          core_physical=$(($i + $number_core_physical))
51          disable_enable 0 $i
52          disable_enable 0 $core_physical
53          i=$(($i - 1))
54        done
55      ;;
56      * ) help=1 ;;
57    esac
58  }
59
60  while [ $quit -eq 0 ]
61   do
62
63    if [ $help -eq 1 ]; then
64      echo "[on|off] [begin|end] [Integer]"
65      help=0
66    fi
67
68    read -p "Enter your choice:" apply mode number
69
70    case $apply in
71      quit ) quit=1 ;;
72      help ) help=1 ;;
73      on ) on_cpu $mode $number;;
74      off ) off_cpu $mode $number;;
75      cpu )
76        CPU=$(cat /proc/cpuinfo | grep "processor")
77        echo $CPU
78      ;;
79      * ) help=1 ;;
80    esac
81   done
82
```

Listing 3.5 – Script pour activer ou désactiver des CPU

Bibliographie

[1] [linux] créer une charge cpu / load. https ://www.djerfy.com/linux-creer-une-charge-cpu-load/.

[2] Bash comment mettre en veille ou retarder un laps de temps spécifié. https ://www.cyberciti.biz/faq/linux-unix-sleep-bash-scripting/.

[3] Tuyêt Trâm DANG NGOC. *Pogrammation système*, chapter "Scripts shell, Flux et filtres". Univérsité de Cergy Pontoise, Cergy, FR, 2018.

[4] Eric Sanchis. *Introduction à la programmation en Bash*. Univérsité de Toulouse 1 Capitole, Toulouse, FR, 2015.

[5] Colin Kelley Thomas Williams. *An Interactive Plotting Program*. 2017.