École d'Automne IA²-2025 – IA et limites planétaires

TP de mesure appliqué à l'IA (mais pas que)

Guillaume Raffin
LIG, INRIA, MIAI Grenoble-Alpes, Bull SAS
guillaume.raffin@univ-grenoble-alpes.fr

Table des matières

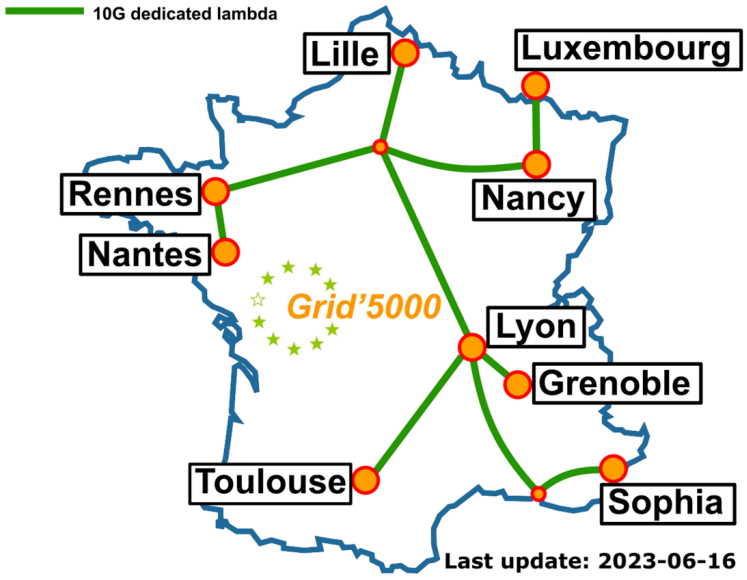
1. Introduction

Cette séance de travaux pratiques a pour objectif de vous faire appréhender la mesure de la consommation énergétique du logiciel et l'estimation de son empreinte carbone en phase d'usage. Vous aurez l'occasion de réaliser des expériences avec un modèle d'IA simple, de mesurer sa consommation et d'estimer son empreinte carbone dans différents scénarios.

Si vous disposez d'un ordinateur sous Linux, vous pouvez réaliser les expériences de mesure interne directement sur votre machine. Cependant, pour la mesure externe, Grid5000 sera utile.

2. Plateforme Grid5000

Nous travaillerons sur Grid'5000, qui est une plateforme expérimentale à grande échelle formée de plusieurs clusters reliés entre eux. Au total, la plateforme compte 8 sites en France, 800 noeuds, et environ 15000 coeurs.



Pour ce TP nous utiliserons le cluster « nova » situé sur le site de Lyon. Afin de vous connecter au cluster, vous aurez beoin d'un client SSH.

Si vous êtes sous Linux ou macOS, vous avez déjà un client SSH.

Sous Windows, essayez d'exécuter `ssh` dans un terminal (CMD, Windows terminal, ...) pour vérifier. Si la commande n'existe pas, installez les outils PuTTY et PuTTYgen.

Sous Windows, il se peut que le service ssh ne soit pas démarré. Pour éviter ce problème, ouvrez une fenêtre PowerShell en mode administrateur et exécutez les commandes suivantes.

```
Get-Service ssh-agent | Set-Service -StartupType Automatic
Start-Service ssh-agent
```

2.1. Premier accès à Grid'5000

Configurons votre accès à la plateforme de calcul.

1. Générer une clé ssh :

```
ssh-keygen
```

1. Ouvrez le courriel que vous avez reçu et cliquez sur le lien de finalisation de compte. Choisissez un mot de passe et entrez la clé SSH que vous venez de générer (copiez-collez le contenu du fichier `.pub`)
2. Une fois votre compte finalisé, connectez-vous avec :

```
ssh login @ access.grid5000.fr
```

Utilisez ici votre login Grid'5000.

Félicitations, vous êtes maintenant connecté au « front-end » de Grid'5000 !

Mais ce serveur ne permet pas d'accéder aux ressources de calcul, il ne sert que de porte d'entrée. Dernière étape : accéder à un site de calcul.

1. Accédez au site de Lyon (depuis le même Terminal) :

```
ssh lyon
```

Si besoin, vous trouverez plus de détails sur le wiki Grid'5000.

2.2. Utilisation du cluster nova

Un nombre suffisant de noeuds du cluster « nova » a déjà été reservé par votre encadrant de TP, sous la forme d'un « container job » qui a pour identifiant `1927880`.

Pour accéder aux ressources de calcul, vous allez créer un job à l'intérieur de ce « container job », ce qui va réserver un noeud du cluster pour vous. Pour cela, demandez un nouveau job au sein du « container job » et pour le projet `lab-2025-ia2-aussois` en lançant la commande suivante :

```
oarsub -t inner=1927880 -l nodes=1,walltime=2 -I --project lab-2025-ia2-aussois -t
'monitor=bmc_cpu_temp_celsius'
```

Explications des options :

- `-t inner=JOB` permet de créer votre job au sein d'un job « container »
- `-l nodes=1,walltime=2` demande 1 noeud pendant 2 heures
- `-I` permet d'obtenir un job interactif, ce qui vous connecte directement au noeud avec une session SSH
- `--project …` spécifie sur quel projet il faut décompter le temps passé à utiliser le noeud
- `-t 'monitor=…'` permet d'activer la mesure externe de certaines informations supplémentaires, ici la température du CPU.

Vous devriez obtenir une sortie similaire à celle-ci :

```
# Set walltime to 3h.
OAR_JOB_ID=1927888
# Interactive mode: waiting...
# Starting...
graffin@nova-19:~$
```

On voit ici l'id de votre nouveau job (1927888), ainsi que le noeud qui vous a été attribué (nova-19).

Note : l'utilisation de la plateforme est soumise à des règles, détaillées ici. Pour résumer très rapidement, vous ne devez utiliser la plate-forme que pour des usages relatifs au TP proposé, pas pour miner du Bitcoin ni pour pirater la messagerie de l'ancien/nouveau premier ministre :)

3. Premières mesures externes

Dans cette section, vous allez visualiser les données de consommation énergétique fournies par les **wattmètres physiques** installés au niveau des alimentations des nœuds de calcul. Ils mesurent la consommation totale du serveur.

Les données des wattmètres sont collectées et exposées par un outil de monitoring appelé « Kwollect », déployé dans Grid'5000. Il collecte et met à disposition un certain nombre de métriques relatives aux nœuds de calcul. Vous pouvez trouver plus de détails sur le monitoring de Grid'5000 avec Kollect dans le wiki Grid'5000.

Vous pouvez visualiser les métriques collectées par Kwollect de deux manières principales : via les tableaux de bord Grafana et en consultant directement l'API Kwollect.

3.1. Kwcollect en ligne avec Grafana

Grid'5000 propose des tableaux de bord Grafana afin de visualiser les métriques disponibles dans Kwollect pour l'ensemble des noeuds. Vous pouvez consulter les tableaux de bord du site Lyon en suivant ce lien.

Pour visualiser les données des wattmètres physiques vous devez sélectionner le noeud dans l'onglet device (par exemple `nova-19`), la période de temps en haut à droite, et la métrique (par exemple `wattmetre_power_watt` pour la puissance électrique consommée, remontée par les wattmètres externes).

Question : quelle est la consommation moyenne du noeud au repos ?

Note: sur le noeud, utilisez la commande `htop` pour vérifier qu'il est bien au repos (ou quasiment).

Question : À quelle fréquence Kwollect renvoie-t-il la métrique wattmetre_power_watt dans cette période ?

Copiez le graphique obtenu avec `scp` et analysez-le.

4. Premières mesures internes

Pour continuer, expérimentons la mesure interne en utilisant un outil de mesure logiciel directement sur le noeud.

4.1. Installation d'Alumet

Pour ce TP, nous utiliserons le logiciel Alumet, développé conjointemenent par des chercheurs du LIG et des ingénieurs de Bull.

Récupérez l'outil depuis le répertoire de votre intervenant.

```
wget http://public.lyon.grid5000.fr/~graffin/alumet-agent
```

Assurez-vous qu'il puisse être exécuté :

```
chmod +x alumet-agent
```

Vérifiez qu'il peut se lancer :

```
./alumet-agent --version
```

4.2. Mesure avec Alumet

Alumet est un logiciel de mesure modulaire qui propose de multiples plugins, que vous pouvez choisir d'activer ou non. Cette modularité permet de s'adapter aux différents environnements matériels et logiciels.

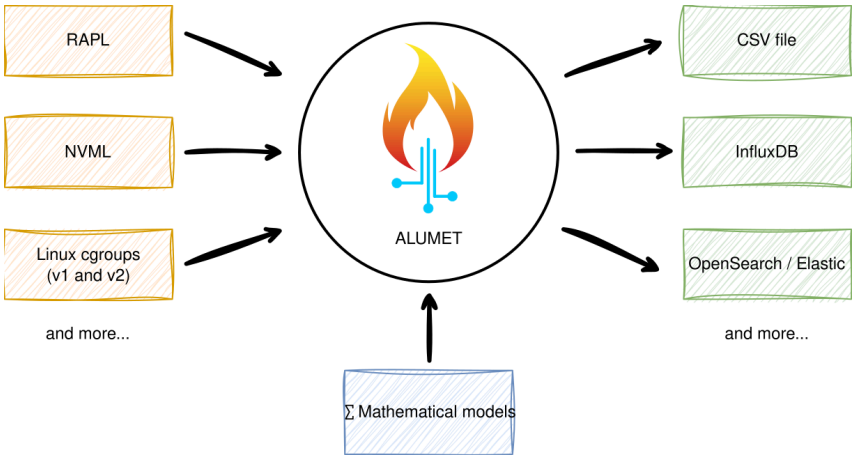


Fig. 1. – Principe de fonctionnement d'Alumet avec ses plugins

Pour le moment, lancez Alumet avec les plugins `rapl` et `csv` et une nouvelle configuration (qui sera générée automatiquement).

```
./alumet-agent --plugins rapl,csv
```

Question : pourquoi ça ne fonctionne pas ?

Pour des raisons de sécurité et de confidentialité, l'accès à RAPL est bloqué par défaut. C'est l'une des difficultés dans la réalisation d'un outil de mesure : les informations dont on a besoin peuvent voir leur accès restreint.

Pour le débloquer, obtenez d'abord le droit d'utiliser `sudo` avec une commande spécifique à Grid5000 :

```
sudo-g5k
```

Puis autorisez les accès aux compteurs de performance à toutes les applications :

```
sudo sysctl -w kernel.perf_event_paranoid=0
```

Relancez.

Deux fichiers sont créés :

- `alumet-config.toml` qui contient la configuration de l'outil de mesure
- `alumet-output.csv` qui contient les valeurs mesurées par l'outil

Le fichier de sortie est mis à jour en continu, au fur et à mesure que de nouvelles mesures sont réalisées par l'outil. Pour l'arrêter, faites Ctrl+C.

Question : Quelle est la fréquence d'acquisition par défaut du plugin RAPL ?

Faites tourner l'outil de mesure quelques temps et observez la sortie (les messages affichés) et le fichier CSV.

Quelles sont les métriques disponibles avec cette configuration ? Où se trouve la consommation d'énergie du CPU ? Avec quelle unité est-elle rapportée ?

La technologie RAPL (Running Average Power Limit) permet de mesurer la consommation de plusieurs parties du CPU et de certains composants qui lui sont reliés. Chaque zone pour laquelle RAPL fournit une donnée est appelée un « domaine ». La liste des domaines disponibles dépend du modèle de CPU et du cablage de la machine (ici votre propre laptop ou l'un des serveurs du cluster nova de Grid5000). La Fig. 2 illustre les différents domaines possibles et ce qu'ils mesurent. Le nom des domaines est en *italique*.

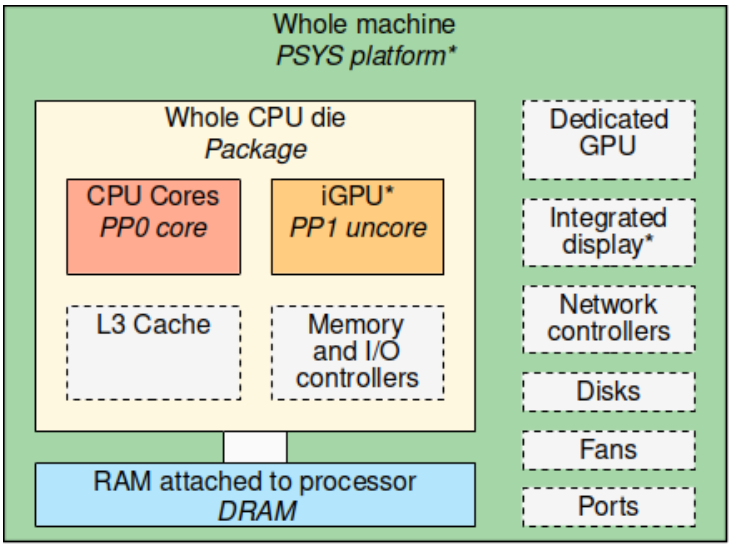


Fig. 2. – Cartographie des domaines RAPL existants [1]

Quels sont les domaines disponibles sur votre machine / sur le serveur Grid5000 ?

4.3. Visualisation des mesures

Pour visualiser les mesures, le plus simple est de faire coopérer Alumet et Kwollect, afin de visualiser les données dans Grafana. Pour cela, utilisez le plugin `kwollect-output`.

Générez une nouvelle configuration avec ce plugin. Les réglages par défaut devraient fonctionner directement.

```
./alumet-agent --plugins rapl,kwollect-output config regen
```

Puis relancez Alumet.

```
./alumet-agent
```

Ouvrez le dashboard et cherchez la métrique `rapl_consumed_energy`, fournie par Alumet.

Fig. 3. – Dashboard avec trois sources de mesure

RAPL remonte la consommation d'énergie par petits intervalles de temps. On peut remonter à la puissance par un petit calcul (1 Joule = 1 Watt seconde). À partir de deux mesures successives fournies par Alumet, **calculez la puissance consommée par le package de votre CPU pendant qu'il ne fait rien** (à part faire tourner SSH et l'outil de mesure).

Modifiez le fichier de configuration pour changer la fréquence d'acquisition. La valeur à modifier est `poll_interval` dans la section correspondant au plugin RAPL. Elle définit l'intervalle de temps entre deux mesures effectuées par l'outil. Alumet peut fonctionner sur une plage de fréquences très large, nous l'avons par exemple testé à 10kHz (intervalle de `0.1ms`). **Attention, vous ne pourrez pas envoyer 10kHz de mesure dans Kwollect.**

Observez l'impact de la mesure à haute fréquence sur la consommation des différents domaines RAPL.

4.4. Écart entre mesure interne et mesure externe

Quel est l'écart entre les valeurs rapportées par les deux familles de mesures ? On peut également se demander si cet écart est constant ou s'il évolue en fonction des expériences. Vous pourrez revenir à cette question à la fin du TP.

5. Empreinte carbone d'un modèle d'IA

5.1. Installation des dépendances

Avant de procéder aux expériences, il faut installer quelques dépendances. Nous allons utiliser l'outil `uv` pour obtenir une version récente de python et installer certains paquets.

```
wget -qO- https://astral.sh/uv/install.sh | sh
uv python install 3.10.12
uv venv
source .venv/bin/activate
uv pip install matplotlib scikit-learn keras tensorflow joblib pandas numpy
```

5.2. Présentation des modèles

Pour ce TP, nous allons utiliser des IA basées sur des réseaux de neurones, plus précisément sur une architecture dite « Multi-Layer Perceptron ». Le but de l'exercice n'étant pas de concevoir une IA

avancée type LLM très performant, nous allons manipuler des IA qui effectuent des tâches relativement simples.

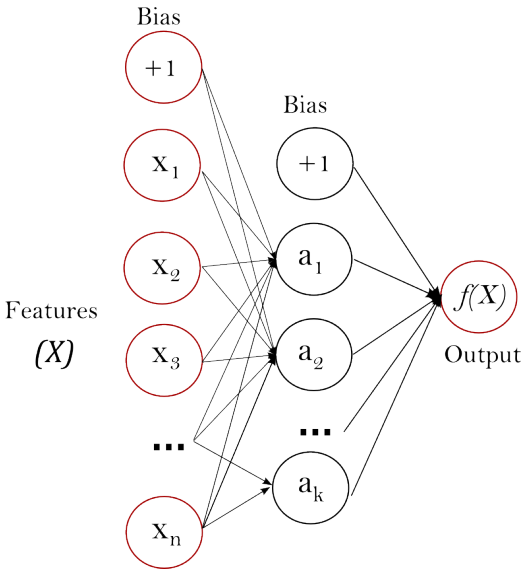


Fig. 4. – Exemples de Multi-Layer Perceptron avec une couche d'entrée à gauche, une couche profonde ou « couche cachée » au centre et une couche de sortie à droite (extrait de la doc SciKit-Learn)

Le premier modèle est un petit réseau de neurones avec une seule couche profonde. On va l'entraîner à classer des points 2D en différents groupes. Les données d'entraînement et de test seront générées artificiellement.

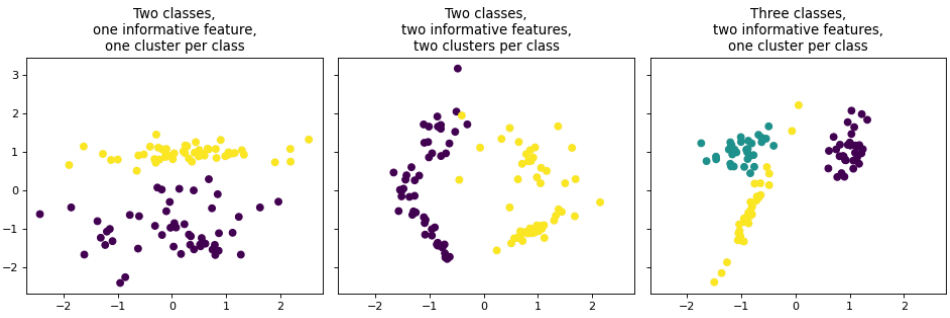


Fig. 5. – Exemples de jeux de données générés pour entraîner ce premier modèle, avec les groupes en couleur (extrait de la doc SciKit-Learn)

Pour l'entraîner, lancer le script python `ai_classifier_training.py`. Une fois l'entrainement terminé, le modèle (donc l'architecture du réseau et les poids des neurones) est sauvegardé dans un fichier. Avec ce setup, on peut mesurer la consommation de l'entraînement et de l'inférence séparément.

```
python3 ai_classifier_training.py
python3 ai_classifier_inference.py
```

Le deuxième modèle est un réseau de neurones plus grand avec de multiples couches. On va l'entraîner à reconnaître des images dans le dataset CIFAR10 (32x32 pixels). Les images sont petites mais, comme nous n'utilisons pas de GPU, l'entraînement prendra quand même jusqu'à 5 minutes.

Fig. 6. – Exemples d'images dans le jeu de données CIFAR10

Le script `ai_img_recog` télécharge le jeu de données, entraîne le modèle sur quelques images et le teste sur le reste des images.

```
python3 ai_img_recog.py
```

Le temps est affiché avant/après les différentes étapes afin de vous permettre de « découper » les mesures de consommation énergétique et de les assigner aux différentes phases du programme. C'est un setup un peu moins facile à exploiter, mais qui reflète certaines difficultés que l'on rencontre lors de l'évaluation de l'empreinte d'un véritable système, où les différentes phases ne sont pas toujours testables de manière indépendante.

5.3. Consommation électrique

5.3.1. Modèle 1 (points 2D)

Pour faciliter la mesure, Alumet offre une commande `exec` qui permet de lier l'outil de mesure à l'exécution d'un programme tiers. Le mode `exec` va démarrer la mesure juste avant le lancement du programme, lancer le programmen, mesurer en continu pendant son exécution, prendre une dernière mesure juste après son arrêt, puis stopper Alumet.

```
alumet-agent --config alumet-config.toml --plugins rapl,csv exec python3
ai_classifier_training.py
```

Utiliser le script `plot_alumet_rapl.py` ou la méthode de votre choix pour visualiser les données du fichier CSV et obtenir la consommation totale de l'entraînement.

Faire la même chose avec l'inférence (lancée par `python3 ai_classifier_inference.py`).

5.3.2. Modèle 2 (CIFAR10)

Pour le deuxième modèle, `exec` peut aussi servir, mais ce sera à vous de retrouver où commencent les différentes phases du programme :

- téléchargement des images
- entraînement du modèle d'IA
- validation par inférence
- affichage de quelques informations

5.4. Empreinte carbone

Pour calculer l'empreinte carbone de l'électricité consommée, il faut connaître son intensité carbone (gCO2_eq / kWh).

RTE fournit cette information sur son site, ou à travers des agrégateurs de données tels qu'ElectricityMaps.

Calculer l'empreinte carbone de l'entrainement et de l'inférence des modèles testés. Assurez-vous que l'énergie est dans la bonne unité (1 Joule = 2,78 × 10^−7 kWh).

Extrapoler à un déploiement massif du modèle : que se passerait-il si on effectuait 100 000 inférences par jour, 7j/7 pendant un an ?

Refaites le calcul en partant du principe que l'expérience a eu lieu en Pologne et non plus en France.

6. Pour aller plus loin

6.1. API Kwollect

Kwollect met a disposition l'ensemble des métriques via une API web. Vous pouvez récupérer les données de consommation d'un noeud directement depuis l'API.

Afin de faciliter la récupération des données de consommation depuis l'API Kwollect et leur visualisation, nous avons récupéré quelques scripts préparés par Vladimir Ostapenco et Laurent Lefèvre.

Depuis le cluster de Lyon, récupérez et ouvrez l'archive contenant les scripts.

```
wget http://public.lyon.grid5000.fr/~graffin/scripts_front.tar.gz
tar -xvf scripts_front.tar.gz
```

Utilisez le script pour retrouver les données concernant votre utilisation du noeud (celles montrées par Grafana).

```
python3 get_consumption_kwollect.py --site lyon --host NOM_DU_NOEUD --start
[DATE_DE_DEBUT] --end [DATE_DE_FIN] --csv kwollect_idle.csv
```

Les dates doivent être fournies au format représenté par l'exemple suivant : `2024-05-30T10:52:30`.

Ce script récupérera les données de consommation du nœud entre deux dates et les enregistrera dans un fichier CSV.

Vous pouvez le copier sur votre machine locale avec la commande scp (dans un autre terminal, local cette fois).

```
scp <login>@access.grid5000.fr:lyon/kwollect_idle.csv  .
```

Analysez le contenu de ce fichier. Combien de lignes contient-il ?

Utilisez le script `plot_consumption_kwollect.py` pour créer un graphique de consommation à partir du fichier CSV.

```
python3 plot_consumption_kwollect.py --csv kwollect_idle.csv --plot
kwollect_plot.jpg
```

6.2. Facteurs de consommation

Pour aller plus loin, nous allons réaliser différentes expériences permettant de mettre en évidence l'impact de certains facteurs sur la consommation électrique de certains composants.

6.2.1. Hyperparamètres du modèle d'IA

Ouvrez le script `ai_img_recog.py` et localisez la configuration du réseau de neurones (avec le `MLPClassifier`).

Faites varier la taille des couches intermédiaires (`hidden_layer_sizes`), enlevez/ajoutez des couches, et relancer le script (avec Alumet). Quel impact sur le temps d'exécution ? Sur la consommation d'énergie instantanée et totale ?

Pour plus de détails sur les différentes options possibles, se référer à la documentation SciKit-Learn.

6.2.2. Température

Plus le matériel est sollicité, plus il consomme d'électricité (dans la mesure de ses capacités, bien sûr) et plus il chauffe. La température n'est pas qu'une conséquence de la consommation, elle a une influence sur celle-ci. En effet, une augmentation de température augmente, entre autres, les courants de fuite du circuits. De plus, un composant qui surchauffe va se brider pour préserver son intégrité physique. Un CPU ou GPU peut ainsi se « mettre en pause » temporairement ou réduire sa fréquence le temps que la température redescende à un niveau acceptable. On parle de « throttling ». La température dépend en partie de facteurs externes, tels que la météo.

Pour réaliser des expériences rigoureuses sur le plan scientifique, il est important de neutraliser l'effet de tels facteurs. Pour en apprendre plus sur la science reproductible, vous pouvez vous tourner vers le MOOC INRIA « Recherche reproductible ».

Vous pouvez visualiser la température dans le Grafana de grid5000 en sélectionnant les métriques qui contiennent le terme `temp`.

Quelle est la température du CPU au repos ? Et pendant l'entraînement ?

6.2.3. Charge

Dans cette section, nous allons effectuer une séquence de tests de charge et de configurations pour comprendre l'impact de la charge CPU sur la consommation d'un nœud de calcul.

6.2.3.1. Question préliminaire : combien de coeurs sont disponibles sur le noeud ?

Pour répondre à cette question, utilisez la commande `lscpu`. Vous obtiendrez, parmi toutes les informations disponibles, le modèle du processeur (quelque chose comme `Intel Xeon CPU E5-2620`) et un nombre noté `CPU(s)`. **Mais à quoi correspond-il vraiment ?**

Utilisez ensuite la description des nœuds Grid'5000 disponible ici. Trouvez le nombre de cœurs physiques disponibles à partir du modèle de CPU. (Note : Les noeuds du cluster nova ont 2 processeurs physiques). Y a-t-il une différence entre le nombre de cœurs donnés par commande lscpu et le nombre de cœurs réellement disponibles sur les CPU ?

6.2.3.2. Différents niveaux de stress

Dans cette section, vous exécuterez l'outil stress avec le nombre différent de workers allant de 1 à 32. Vous ferez une pause d'environ dix secondes entre les exécutions.

Pour réaliser cette expérience, vous pouvez automatiser les exécutions à l'aide d'un script bash comme suit.

```
#!/bin/bash
START_DATE=$(date +%Y-%m-%dT%H:%M:%S)
echo "Sleeping for 10 seconds before test."
sleep 10
for val in 1 2 4 8 16 32; do
echo "Launching stress with $val workers for 20 seconds."
stress -c $val -t 20
echo "Sleeping for 10 seconds."
sleep 10
done
END_DATE=$(date +%Y-%m-%dT%H:%M:%S)
echo "EXPERIMENT START DATE: $START_DATE"
echo "EXPERIMENT END DATE: $END_DATE"
```

Enregistrez ce contenu dans un fichier texte avec l'extension .sh (par example stress_cpu_num.sh), rendez le fichier exécutable avec la commande chmod et lancez-le.

Note : pour rendre le fichier exécutable, utilisez la commande `chmod +x stress_cpu_num.sh`.

Notez les dates de début (EXPERIMENT START DATE) et de fin (EXPERIMENT END DATE) de l'expérience. Elles seront utilisées pour l'exécution du script `get_consumption_kwollect.py`.

Récupérez et visualisez la consommation du nœud avec les scripts `get_consumption_kwollect.py` et `plot_consumption_kwollect.py`. Utilisez la commande scp vue précédemment pour copier le graphe de consommation sur votre machine.

Questions :
- Que se passe-t-il lors des dernières exécutions de stress ?
- Quelle est la consommation moyenne avec chaque nombre de workers ?
- Donnez une valeur approximative. Quelle est la puissance par worker ? Que ne voit-on pas sur ce graphe ?
- Pourquoi faire une pause entre chaque stress ?

6.3. Variation de fréquence

Dans cette section, vous allez modifier la fréquence du processeur et voir comment cela affecte la consommation énergétique du nœud de calcul.

Depuis le cluster de Lyon, récupérez et ouvrez l'archive contenant de nouveaux scripts.

```
wget http://public.lyon.grid5000.fr/~graffin/scripts_node.tar.gz
tar -xvf scripts_node.tar.gz
```

Visualisez les fréquences disponibles avec la commande

```
cpupower frequency-info.
```

Questions :
- Quelle est la fréquence minimale supportée par le processeur ?
- Quelle est la fréquence maximale supportée par le processeur ?

Pour changer la fréquence du CPU vous allez utiliser le script `change_cpu_freq.sh`. Il prend un argument `CPU_FREQ_VALUE`, qui est la fréquence de CPU souhaitée (par exemple 1.6Ghz). Il nécessite des privilèges pour s'exécuter, vous devez le lancer avec `sudo-g5k` (Grid5000 possède sa propre version de `sudo`, avec certaines contraintes).

Déroulement de l'expérience :

- Notez le temps du début
- attendez 10 secondes
- changez la fréquence du CPU à la fréquence minimale
- attendez 10 secondes
- exécutez un stress pendant 20 secondes avec 16 workers
- attendez 10 secondes
- notez le temps de fin
- réinitialisez la configuration du gestionnaire de fréquences CPU avec le script `restore_cpu_governor.sh`

Vous pouvez automatiser cette expérience à l'aide d'un script bash :

```
#!/bin/bash
START_DATE=$(date +%Y-%m-%dT%H:%M:%S)
echo "Sleeping for 10 seconds before test."
sleep 10
sudo-g5k ./change_cpu_freq.sh <CPU_FREQ_VALUE>
sleep 10
stress -c 16 -t 20
sleep 10
END_DATE=$(date +%Y-%m-%dT%H:%M:%S)
echo "Restore CPU Governor after experience."
sudo-g5k ./restore_cpu_governor.sh
echo "EXPERIMENT START DATE: $START_DATE"
echo "EXPERIMENT END DATE: $END_DATE"
```

Commencez par mettre la fréquence du CPU minimale au lieu de `<CPU_FREQ_VALUE>`. Vous pouvez ensuite refaire l'expérience avec une fréquence intermédiaire (en supposant qu'elle soit supportée), puis avec la fréquence maximale. Attention à bien préciser l'unité dans la valeur, par exemple `2Ghz`.

Notez les dates de début (EXPERIMENT START DATE) et de fin (EXPERIMENT END DATE) de l'expérience. Ils seront utilisées lors de l'exécution du script `get_consumption_kwollect.py` (ou lors de la lecture sur le Grafana).

Questions :

- Comment évolue la consommation lors de ce stress ? Et au repos (dans les pauses autour du stress) ? Utilisez le Grafana ou les scripts `get_consumption_kwollect.py` et `plot_consumption_kwollect.py` vus précédemment.
- Quelle différence avec le stress précédent dans laquelle la fréquence du CPU n'était pas limitée par nos soins ?

Bonus : lancez le script en arrière plan et regardez ce que retourne la commande `cpupower frequency-info`.

Bibliographie

[1] G. Raffin et D. Trystram, « Dissecting the Software-Based Measurement of CPU Energy Consumption: A Comparative Analysis », *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, nº 1, p. 96-107, 2025, doi: 10.1109/TPDS.2024.3492336.