TP InfoResp
Mesures énergétiques – partie 3

25 septembre 2024

Les objectifs de ce TP sont les suivants :
— Tester la performance énergétique d'une plateforme de calcul en fonction de la fréquence processeur utilisée.
— Comparer les performances d'implémentations dans différents langages d'un même algorithme.

1 Effet de la fréquence

L'objectif est ici d'analyser l'effet de la fréquence du processeur en terme d'efficience énergétique : vaut-il mieux utiliser une fréquence élevée pour éteindre au plus tôt un nœud, ou l'inverse ?

Pour cela, on va appeler `sysbench` au sein d'un script Python en spécifiant une quantité de calcul fixée et en augmentant le nombre de threads de manière à solliciter **tous** les cœurs de calcul :

```
os.system("sysbench --test=cpu --threads=1 --events=100000 --time=0 run")
```

Penser à ajuster le nombre d'événements pour que la charge dure entre 5 et 20 secondes de manière à pouvoir éviter les effets de démarrage (si ça ne dure pas assez longtemps), mais aussi de devoir attendre trop longtemps (dans le cas inverse).

On réalisera un script qui entoure cet appel du code nécessaire pour calculer l'énergie consommée à partir de l'interface REST de Grid5000. En complément, on pourra aussi lancer le script avec likwid pour avoir la mesure fournie par RAPL qui complétera la mesure réalisée par le wattmètre.

En ce qui concerne les fréquences, on identifiera d'abord les fréquences disponibles avec `lscpu`. Il est possible de fixer une fréquence unique pour tous les cœurs d'un nœud ainsi :

```
noeud$ sudo-g5k cpupower frequency-set -g performance
noeud$ sudo-g5k cpupower frequency-set -u 1000MHz
```

Mesurer l'énergie avec les wattmètres et RAPL pour une charge processeur constante avec une dizaine de fréquences repartie entre la fréquence minimale et celle maximale. En tirer une courbe qui montre l'effet de la fréquence sur l'énergie.

```
import requests
import datetime

site = "lyon"
metric = "wattmetre_power_watt"
node = "taurus-10"
start = datetime.datetime.now().replace(microsecond=0).isoformat()
# some computations
stop = datetime.datetime.now().replace(microsecond=0).isoformat()
url = "https://api.grid5000.fr/stable/sites/%s/metrics?metrics=%s&nodes=%s&
    start_time=%s&end_time=%s" \
            % (site, metric, node, start, stop)
```

```
data = requests.get(url, verify=False).json()
sum(item['value'] for item in data) / len(data)
```

Il est fréquent qu'un compromis soit la meilleure solution d'un point de vue énergétique. En effet, si le processeur est trop lent, le processeur ne consommera pas beaucoup mais le reste du nœud aura une consommation fixe (dite statique) qui dominera dans l'énergie consommée. À l'inverse, les fréquences élevées sont généralement moins efficientes.

2 Ensemble de Mandelbrot

L'ensemble de Mandelbrot est une célèbre fractale. Nous allons étudier les performances d'implémentations permettant de le calculer et de générer une image qui le représente.

Récupérer et lancer les 2 implémentations en Java et en Python. Pour chacune, 2 arguments sont nécessaires : le paramètre dont dépend la complexité des calculs (de 100 à 10000) et le fichier image de sortie (`.bmp`).

Après avoir tester les 2 programmes, réaliser un script qui permet de lancer chacune des implémentations pour différentes valeurs de paramètre. Compléter le script d'exécution pour récupérer les mesures des wattmètres, puis calculer l'énergie consommée pour chaque paramètre et chaque implémentation.

Tracer une courbe montrant l'effet du paramètre et du langage d'implémentation sur l'énergie consommée.