Mesure et analyse de la consommation d'énergie électrique des programmes

Sid Touati
Professeur à l'université Côte d'Azur

Le but de ces travaux pratiques est de vous initier à la mesure de consommation d'énergie électrique d'un programme avec un outil appelé `EcoFloc`. Afin de pouvoir faire ce TP, vous devez installer les logiciels suivants sur votre machine personnelle :

1. Linux (64bits).
2. Compilateurs C : `gcc` (GNU C compiler) et `icc/icx` (Intel C Compiler).
3. `EcoFloc` : `https://github.com/labDomolandes/ecofloc`

1 Installation et prise en main

Compilez et installez l'outil EcoFloc sur votre machine. Lisez la documentation de prise en main et faites quelques tests d'initiation.

Dans une première version de cet outil, il est nécessaire de lancer manuellement le processus à évaluer, détecter son PID, puis configurer EcoFloc pour analyser sa consommation. En d'autres terme l'outil évalue la consommation d'énergie de la part d'un processus déjà lancé. Ceci laisse présager que l'outil est plus adapté aux processus longs en tache de fond. Si un processus a un début et une fin, l'outil ne permet pas d'évaluer sur toute la durée d'exécution comme le ferait la commande `time` par exemple. Lancer la commande `ecofloc` en plein milieu d'exécution d'un processus qui tourne veut dire qu'on a raté le début de l'exécution. Une solution à cela serait d'inclure un point de synchronisation dans votre programme à évaluer, par exemple une lecture d'un caractère du clavier. Cela interromprait le processus en attente d'une touche clavier, vous permettant ainsi de lancer la commande `ecofloc` à cet instant.

EcoFloc termine son exécution lorsque l'application à évaluer est terminée. Pour ça, il faut configurer une valeur de temps négative, permettant à EcoFloc de s'exécuter sans limite de durée.

2 Micro-benchmarks *CPU-bound*

Dans cet exercice, il vous est demandé de programmer un micro-benchmarks qui ne fait que des calculs flottants sur des variables scalaires uniquement, pas de tableaux. Le calcul soit se faire majoritairement sur le CPU. Ce type de programmes sont qualifiés de *CPU-bound* car leur vitesse d'exécution dépend principalement des performances du CPU. Ci-dessous un exemple de pseudo-code *CPU-bound*.

```
#define N 100000
// modifiez la valeur de N selon vos besoins
float x,y,z,t;
int i;
for (i=0; i< N; i++){
  x=x+y;
  y=x+t;
  z=x+z;
```

```
  x=t+y;
  t=y+x;
}
// il faut imprimer le résultat du calcul sinon le compilateur supprimera ce code
```

- Générez plusieurs versions binaires en utilisant les deux compilateurs `gcc` et `icx`, en utilisant les différentes options `-O0, -O1, -O2, -O3, -Os`.
- Pour chaque version binaire, évaluez le temps d'exécution CPU et la consommation d'énergie électrique avec l'outil `EcoFloc`. Faites plusieurs répétitions et sauvegardez les résultats.

3 Micro-benchmarks *Memory-bound*

Dans cet exercice, il vous est demandé de programmer un micro-benchmarks qui ne fait que des calculs sur des tableaux en mémoire. Ce type de programmes sont qualifiés de *Memory-bound* car leur vitesse d'exécution dépend principalement des performances d'accès à la mémoire. Ci-dessous un exemple de pseudo-code *Memory-bound*.

```
#define Taille 1000000
// modifiez les valeurs de Taille selon votre besoin
#define pas 8
float x, tab[N];
int i,j;
for (i=0; i< Taille; i++)  tab[i]= random(); // initialisation
for (i=0; i< Taille; i=i+pas)  x=x+tab[i];
// il faut imprimer le résultat x du calcul sinon le compilateur supprimera ce code
```

- Générez plusieurs versions binaires en utilisant les deux compilateurs `gcc` et `icx`, en utilisant les différentes options `-O0, -O1, -O2, -O3, -Os`.
- Pour chaque version binaire, évaluez le temps d'exécution CPU et la consommation d'énergie électrique avec l'outil `EcoFloc`. Faites plusieurs répétitions et sauvegardez les résultats.

4 Micro-benchmarks avec des appels système

Reprenez un micro-bvenchmark précédent et insérez-y des appels systèmes de lecture/écriture dans un fichier. Ci-dessous u : n exemple de pseudo-code

```
#define Taille 1000000
// modifiez les valeurs de Taille selon votre besoin
#define pas 8
float x, tab[N];
int i,j;
for (i=0; i< Taille; i++)  tab[i]= lecture_depuis_fichier(); // initialisation
for (i=0; i< Taille; i=i+pas)  {
   x=x+tab[i];
   ecrire_fichier(x);
 }
```

- Générez plusieurs versions binaires en utilisant les deux compilateurs `gcc` et `icx`, en utilisant les différentes options `-O0, -O1, -O2, -O3, -Os`.
- Pour chaque version binaire, évaluez le temps d'exécution CPU et la consommation d'énergie électrique avec l'outil `EcoFloc`. Faites plusieurs répétitions et sauvegardez les résultats.

5 Analyse des résultats expérimentaux

L'objectif de cet exercice est d'utiliser le logiciel R pour répondre aux questions suivantes, en vous appuyant sur les données expérimentales récoltées dans les exercices précédents :

- Est ce que vous déduisez qu'un compilateur est plus efficace que l'autre en terme de consommation d'énergie ? ou en terme de temps d'exécution ?
- Observez vous une corrélation statistique entre la consommation d'énergie et la vitesse d'exécution d'un programme ? (quelque soit le compilateur)
- Est ce que vos conclusions changent selon si on distingue les micro-benchmarks *CPU-bound* vs *Memory-bound* ?