Analyse de l'Impact Énergétique de Notre Projet

Introduction

L'évaluation de la consommation énergétique est devenue un enjeu clé dans le développement de logiciels modernes. Dans ce document, nous présentons les moyens mis en œuvre pour mesurer l'impact énergétique de notre projet, discutons des implications de nos choix techniques, et analysons les stratégies adoptées pour minimiser cet impact tout en maintenant la qualité et la fiabilité de notre produit.

---

1. Moyens mis en œuvre pour évaluer la consommation énergétique

1.1 Powertop

Nous avons utilisé l'outil Linux **powertop**, qui permet de surveiller la consommation énergétique globale d'un PC.

Fonctionnement :

- Powertop évalue la consommation énergétique par processus en s'appuyant sur des informations fournies par le matériel et les pilotes.
- Il identifie les processus énergivores et fournit des recommandations pour réduire la consommation.

Utilité :

- Powertop nous a permis de repérer les périodes d'intensité énergétique accrue et de les corréler à des étapes spécifiques de notre compilation.

1.2 Commande `/usr/bin/time`

La commande **/usr/bin/time** a été utilisée pour mesurer :

- **Le temps réel écoulé** (real) : une approximation de la durée totale d'exécution d'une commande.
- **Le temps CPU utilisateur** (user) : les cycles CPU utilisés pour exécuter les instructions du programme.
- **Le temps CPU système** (sys) : les cycles utilisés pour gérer les appels système.

Utilité :

- Ces métriques nous ont permis d'établir des corrélations entre les durées d'exécution et la consommation énergétique.

1.3 Fichier Intel RAPL

Le fichier système **/sys/class/powercap/intel-rapl/intel-rapl:0/energy_uj** fournit des informations précises sur l'énergie consommée en microjoules (µJ).

Script de Mesure Énergétique

Nous avons développé un script, **conso**, pour automatiser la mesure énergétique :

```
#!/bin/bash

# Vérifier si un script a été passé en argument
if [ -z "$1" ]; then
  echo "Erreur : Aucun script à exécuter fourni."
  exit 1
fi

# Mesurer la consommation énergétique avant
a=$(sudo cat /sys/class/powercap/intel-rapl/intel-rapl:0/energy_uj)

# Exécuter le programme (le script passé en argument, avec ses paramètres)
/usr/bin/time -p "$@" > /dev/null

# Mesurer la consommation énergétique après
b=$(sudo cat /sys/class/powercap/intel-rapl/intel-rapl:0/energy_uj)

# Calculer et afficher la consommation énergétique
echo "Consommation énergétique : $(($b - $a)) µJ"
```

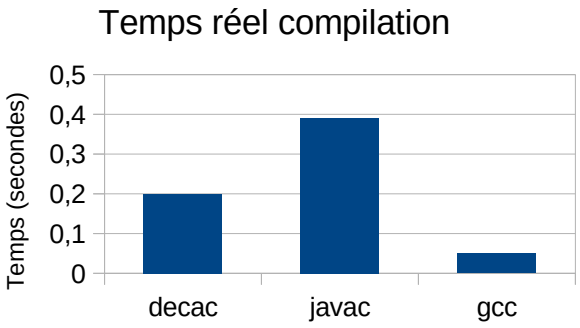
Il suffit de lui passer un programme en argument pour évaluer sa consommation : conso ./mon_programme

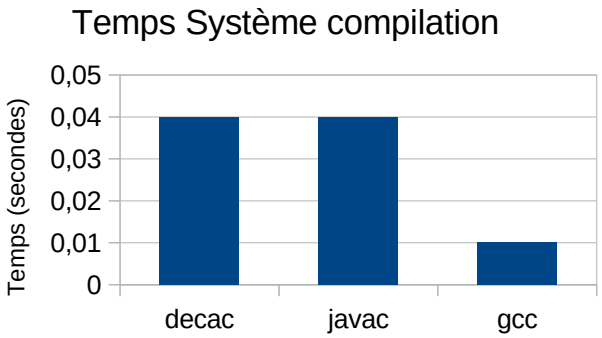
2. Discussion des choix de compilation de Deca vers IMA

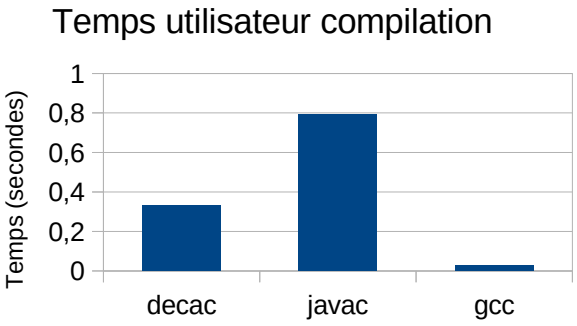
La compilation de Deca vers IMA a été optimisée pour minimiser l'impact énergétique. Les choix suivants ont été effectués :

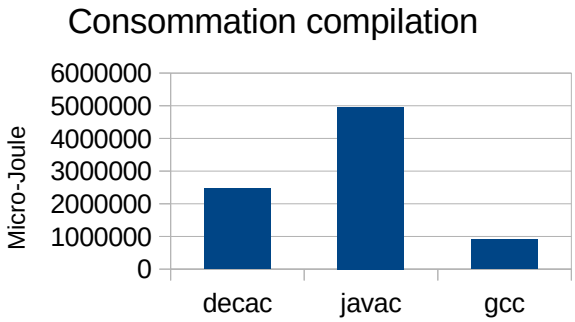
- **Optimisation des structures intermédiaires** : réduction du nombre de passes inutiles.
- **Minimisation des instructions générées** : en limitant la duplication des calculs lors de la génération du code IMA.
- **Rejet des optimisations énergivores** : certaines optimisations gourmandes en CPU ont été jugées superflues pour ce projet.

Nous avons ensuite utilisé le script bash conso, mentionné précédemment, afin de comparer les résultats obtenus avec notre compilateur Deca à ceux de compilateurs largement utilisés tels que javac et gcc. Pour assurer une comparaison équitable, nous avons utilisé des programmes similaires dans chaque langage (C, Java et Deca), consistant en une boucle while exécutant 1 000 000 itérations. Ces programmes ont été compilés, puis exécutés, afin d'analyser leurs performances respectives.

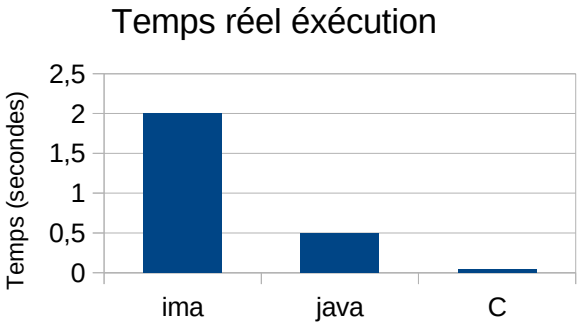


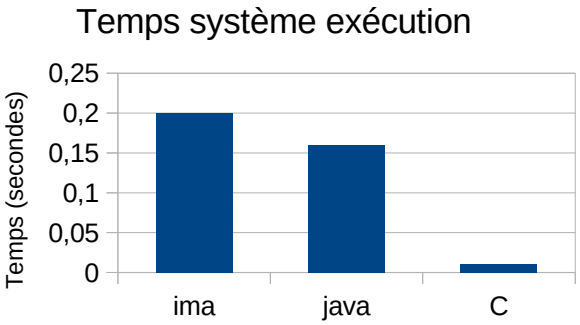


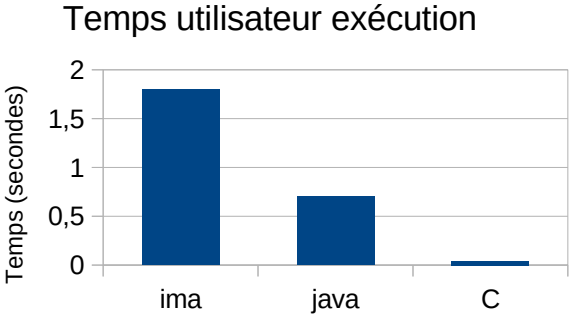


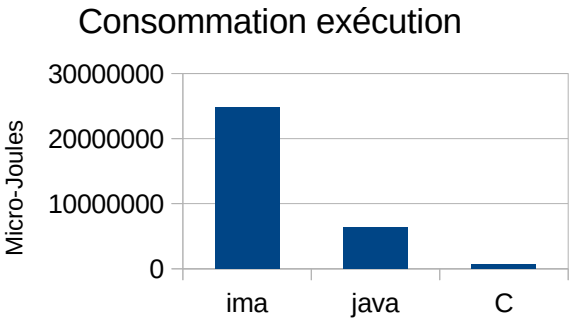


En ce qui concerne la compilation, il est possible de constater que le compilateur javac est moins performant que decac, qui lui-même est moins performant que gcc. En effet, le temps réel de compilation avec javac est deux fois plus élevé que celui de decac, tandis que le temps système reste identique pour les deux. De plus, le temps utilisateur est deux fois plus long avec javac qu'avec decac, et nettement inférieur avec gcc. En ce qui concerne la consommation énergétique, elle est également deux fois plus élevée pour javac que pour decac.









Concernant l'exécution, il apparaît que les programmes compilés avec decac sont moins performants que ceux produits par javac et gcc. En effet, le temps d'exécution réel est de 2 secondes pour Deca, contre 0,5 seconde pour Java et seulement 0,1 seconde pour le langage C. Cette différence se reflète également dans la consommation énergétique, où le programme Deca exécuté via ima consomme quatre fois plus d'énergie que son équivalent en Java.

---

3. Stratégie de validation et réduction de l'impact énergétique

3.1 Mise en cache sous GitLab

Nous avons mis en place un cache dans les pipelines CI/CD pour réduire les téléchargements redondants des dépendances.

Fonctionnement :

- Stockage des dépendances couramment utilisées dans un cache centralisé.
- Réutilisation des données entre différentes exécutions de pipeline.

Résultats :

- Réduction significative des téléchargements réseau.
- Diminution du temps d'exécution et de l'énergie consommée.

3.2 Limitation des tests redondants

Nous avons adopté une stratégie consistant à :

- **Concevoir des tests proches du code** : ciblant directement les fonctionnalités principales.
- **Éliminer les tests inutiles** : suppression de tests répétitifs qui n'apportent pas de valeur ajoutée.

---

4. Impact énergétique de l'extension

Lors de la conception de notre extension, nous avons tenu compte de son impact énergétique en :

- Favorisant des algorithmes à faible complexité.
- Utilisant des structures de données optimisées pour limiter les accès mémoire coûteux.
- Réduisant les dépendances externes non essentielles.

Conclusion

L'effort fourni pour évaluer et réduire l'impact énergétique de notre projet s'inscrit dans une démarche responsable. Grâce à des outils tels que **Powertop**, **/usr/bin/time**, et **Intel RAPL**, nous avons pu identifier des leviers d'optimisation concrets.
Les choix effectués, que ce soit en termes de compilation ou de validation, témoignent d'un équilibre entre performance et efficacité énergétique.