Analyse des impacts énergétiques du projet génie logiciel

Projet Génie Logiciel
Florian ABOU EDOU, Daniel BANNISTER, Loïc GESTIN,
Sarah MEVEL et Tristan SAMINADAYAR

Glossaire

$CO_2 - e$ Selon le GIEC : L'équivalent CO_2 d'une émission de gaz à effet de serre est la quantité de CO_2 qui provoquerait le même forçage radiatif cumulé sur une période donnée, c'est-à-dire qui aurait la même capacité à retenir le rayonnement solaire. Il est exprimé en appliquant un facteur de conversion, le potentiel de réchauffement global (PRG), qui dépend du gaz et de la période considérée.

1 Introduction

L'objectif primordial de ce document est de mener une analyse approfondie de l'empreinte énergétique de notre projet. Pour ce faire, nous négligerons les dépenses énergétiques fixes liées aux bâtiments publics, aux restaurants, au chauffage et au transport, afin de nous focaliser exclusivement sur les aspects numériques de notre initiative. À titre d'illustration, en prenant une consommation moyenne de 7 tonnes de $CO_2 - e$ par habitant en France, la consommation mensuelle du projet pour une équipe de 5 personnes serait d'environ 3 tonnes de $CO_2 - e$.

Dans le cadre de notre projet, trois volets sont identifiés comme étant susceptibles de consommer de l'énergie :

— **Conception du compilateur :** Cet impact découle directement du processus de conception du compilateur, englobant ainsi les coûts énergétiques liés au développement, aux tests et au déploiement.
— **Compilation :** Cette composante représente l'impact énergétique résultant du processus de compilation d'un programme Deca.
— **Exécution :** Ceci englobe l'ensemble des consommations d'énergie liées à l'exécution de programmes assembleurs compilés par notre compilateur, dans notre cas, la consommation de l'`IMA`.

2 Conception du compilateur

2.1 Quelques chiffres

En prenant la moyenne des émissions de CO2 par kWh produit en France sur la période du projet [1], soit environ 40g $CO_2 - e$ par kWh, et en extrapolant la première valeur du tableau pour 20 jours de travail pour 5 personnes, on obtient $99.160 \times 20 \times 5 = 9916\text{Wh} \approx 10\text{KWh}$. Cela équivaut à 400g $CO_2 - e$, soit le même impact qu'un trajet en TGV de 136 km [2] ou l'envoi de 23 courriels [3]. Cependant,

1. Source : RTE - éCO2mix
2. Source : https ://impactco2.fr/transport/tgv
3. Rédiger, envoyer et lire 1 courriel de 1 Mo (pièce jointe) à 5 destinataires via une connexion fixe, stockage pendant 10 ans et 3 redondances pour l'émetteur et le destinataire. Source : Négaoctet - Adème

Type de dépense	Consommation (Wh)	Equivaut à
Une journée de travail sur mon ordinateur portable (consommation globale de la machine)	99.160	Deux recharges complète d'un MacBook Pro
Compilation du projet (3.5s)	0.072	Quasi rien
Execution du script de tests (198.7s)	1.317	Ampoule LED pendant 15 min

TABLE 1 – Les valeurs sont calculées sur mon ordinateur portable à l'aide d'un dongle de mesure sur la source d'alimentation. Les ordinateurs de l'Ensimag, étant des ordinateurs fixes, présentent une consommation significativement plus élevée.

il convient de noter que cette mesure ne prend pas en compte les émissions liées à la fabrication des ordinateurs. En prenant la valeur moyenne de 45.3 kg $CO_2 - e$ par an pour un ordinateur[4], la conception du compilateur a une empreinte carbone de 18.87 kg $CO_2 - e$, équivalente aux émissions moyennes d'une voiture sur 183 km[5].

2.2 Perspectives

La phase de conception du compilateur est en effet consommatrice d'énergie. Mais là n'est pas la dépense la plus importante. En effet, la consommation d'un projet d'un mois à l'Ensimag est assez constante, la vraie économie peut être faite dans les parties suivantes.[6]

3 Compilation et Execution

Dans le scénario où notre compilateur serait largement utilisé, la phase de compilation et d'exécution émergerait nettement comme le secteur le plus gourmand en termes de ressources. La compilation s'effectue généralement à quelques reprises au cours du développement de l'application, tandis que l'exécution peut intervenir de manière très récurrente. À mon sens, il serait plus avisé de concevoir un compilateur plus complexe, donc plus énergivore, mais capable de générer un code considérablement plus optimisé. En effet, à court terme, un compilateur plus gourmand en ressources pourrait engendrer une consommation accrue, mais à long terme, cette surconsommation serait compensée par les gains de performances du programme en utilisation. Cette approche s'inscrit dans une perspective d'optimisation durable, où l'investissement initial en énergie est rentabilisé par une efficacité accrue sur la durée d'utilisation du logiciel.

3.1 Idée d'optimisation

```
1 class A {
2     boolean a(){
3         ... // Un code étendu
4     }
5     boolean b(){
6         ... // Un code étendu
7     }
8     void c(){
9         ... // Un code étendu
```

4. Impact moyen d'un ordinateur portable, incluant la fabrication, le transport et la fin de vie pour un usage professionnel ramené à un an d'utilisation. Source : Négaoctet - Adème

5. Valeur basée sur la moyenne des émissions de CO2 des voitures neuves en 2022, soit 103g $CO_2 - e$ par km. Source : ADEME

6. Nous avons fait le choix de mettre en place un processus de validation via un gilab-CI, qui effectue la compilation et l'ensemble des tests lors des pushs sur notre Git. Il s'agit d'une bonne solution pour s'assurer de la qualité du code produit, mais, en effet, peut avoir des désavantages au niveau de la consommation énergétique du projet.

```
    }
}
{
    int unused_var = 12; // Éviter l'allocation d'espace inutile
    A a = new A();

    if (a.a() && a.b()){
        ... // Évaluer a.b() uniquement si a.a() est vrai
        a.c();
    }
    else if (a.a() || a.b()){
        ... // Évaluer a.b() uniquement si a.a() est faux
        a.c();
    } else {
        a.c();
    }
    // Appeler a.c() 1 fois à la fin de la structure conditionnelle

    if (true) {
        ...
    } else {
        ... // Supprimer le code inutile (mort)
    }
}
```

Listing 1 – Exemple d'optimisations possible en Deca

En outre, plusieurs optimisations peuvent être implémentées au niveau des boucles, telles que la propagation de constante ou l'utilisation d'un garbage collector. Bien que ces optimisations complexifient le processus de compilation, elles contribuent, comme mentionné précédemment, à améliorer l'efficacité du code généré. Bien évidemment, pour avoir un code beaucoup plus optimisé, passer par un interpréteur virtuel de code assembleur augment considérablement le temps d'exécution par rapport à un code assembleur généré selon l'architecture de la machine.

Annexe

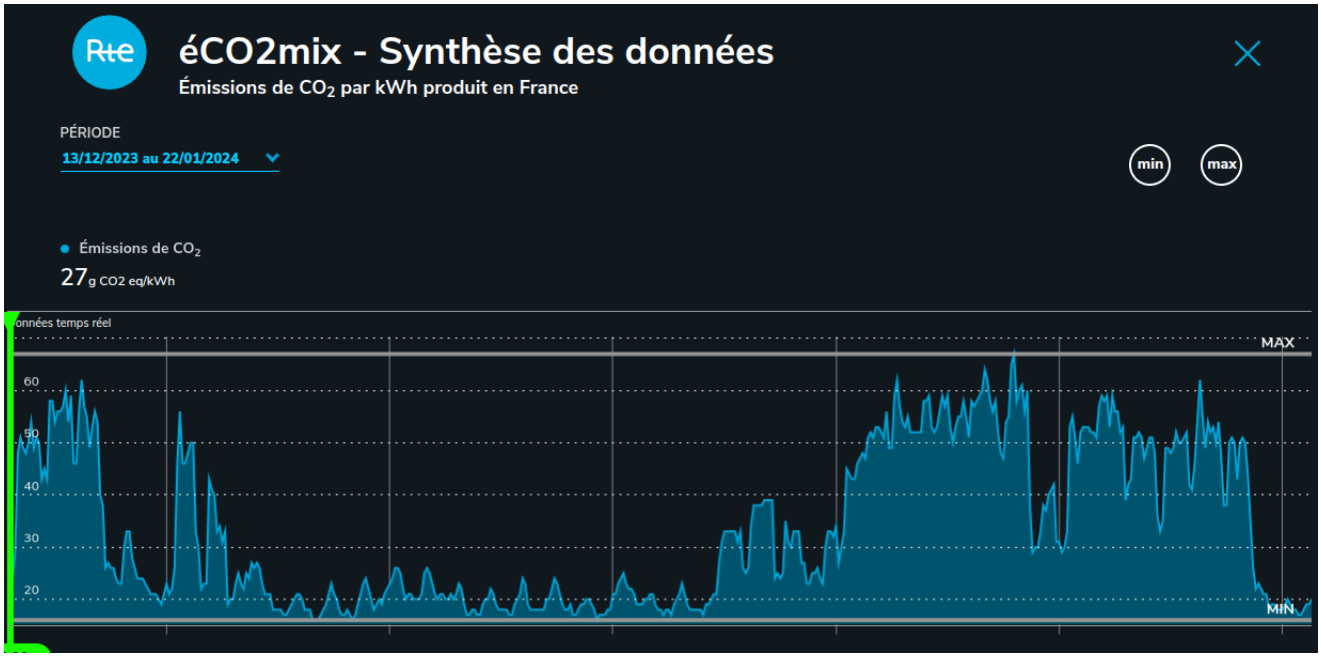

FIGURE 1 – Capture d'écran du site éCO2mix de RTE sur la période 13/12/2023 - 222/01/2024