Analyse approfondie de l'impact énergétique du projet

Projet Génie Logiciel

Groupe : 10

Équipe : 56

Année universitaire : 2025–2026

Table des matières

1 Introduction générale **3**

2 Analyse de l'efficience du code généré **3**

2.1 Méthodologie comparative 3

2.1.1 Application Numérique 3

2.1.2 Interprétation 3

3 Efficience du procédé de fabrication **4**

3.1 Modèle de consommation matériel 4

3.2 Mesures réelles des commandes fréquentes 4

3.3 Validation via Intel RAPL 4

3.3.1 Analyse 4

3.4 Bilan énergétique de la fabrication 5

4 Autres impacts écologiques et usage de l'IA **5**

4.1 Choix du matériel 5

4.2 Numérique vs déplacements 5

4.3 Impact de l'IA 6

4.3.1 Empreinte hydrique 6

5 Conclusion **6**

1 Introduction générale

Le projet de développement du compilateur **Deca** constitue un cadre pertinent pour analyser l'impact énergétique du cycle de développement logiciel. Chaque phase du projet (compilation, tests, validation, exécution future du code) consomme des ressources matérielles et donc de l'énergie.

Cette étude vise à analyser deux dimensions majeures :

— l'efficience du **code produit** par le compilateur ;

— l'efficience du **procédé de fabrication** du projet.

Nous avons combiné des mesures indirectes (temps CPU, cycles) et des mesures physiques via les sondes **Intel RAPL** exposées par le noyau Linux.

2 Analyse de l'efficience du code généré

2.1 Méthodologie comparative

Pour évaluer la performance énergétique intrinsèque de notre compilateur, nous avons mesuré le nombre de cycles processeur (c) générés pour le programme standard *Syracuse* ($u_0 = 42$), comparé à l'état de l'art étudiant ($c_{min} \approx 1346$ cycles).

2.1.1 Application Numérique

Pour le programme de référence `syracuse.deca`, nous obtenons les résultats suivants :

— **Notre compilateur (c)** : $\approx 1\,800$ cycles (Moyenne estimée).

— **Référence optimisée (c_{min})** : $1\,346$ cycles.

L'écart relatif (E) est calculé selon :

$$E = \frac{|1800 - 1346|}{1800} = \frac{454}{1800} \approx 25,2\% \tag{1}$$

2.1.2 Interprétation

Cet écart de **25%** représente notre "dette énergétique logicielle". Concrètement, pour chaque exécution du programme final, notre code consomme un quart de cycles CPU "inutiles" (sauvegardes registres redondantes, sauts non optimisés). Si ce programme était exécuté des millions de fois dans un datacenter, l'impact cumulé dépasserait largement le coût de sa fabrication.

3 Efficience du procédé de fabrication

3.1 Modèle de consommation matériel

Nous supposons une puissance moyenne $P = 15$ W pour un ordinateur portable :

$$E_{elec} = T_{total} \times P$$

3.2 Mesures réelles des commandes fréquentes

Commande	Temps unitaire	Répétitions	Temps Total (s)
mvn clean	3,00 s	200	600
mvn compile	9,29 s	500	4 645
mvn test	16,59 s	200	3 318
mvn verify	18,06 s	50	903
test_all.sh	105,65 s	100	10 565
Total	-	-	**20 031**

TABLE 1 – Temps CPU consommé par les outils de build

3.3 Validation via Intel RAPL

```
$ /usr/bin/time ./src/test/deca/test_all.sh
207.13user 17.75system 1:45.65elapsed 212%CPU

intel-rapl
  Zone 0
    name: package-0
    energy_uj: 8759906163  (MicroJoules)
```

Listing 1 – Log de mesure Powercap sur test_all.sh

3.3.1 Analyse

— **Efficacité parallèle (212% CPU)** : exploitation multi-cœurs réduisant le temps global.

— **Consommation mesurée** : la sonde `package-0` confirme la forte sollicitation CPU.

3.4 Bilan énergétique de la fabrication

En appliquant le modèle sur le temps total ($\approx 5,5$ heures de compilation pure) :

$$20\,031 \text{ s} \times 15 \text{ W} \approx 300\,465 \text{ Joules} \approx 0,083 \text{ kWh}$$

Notre optimisation des tests a permis de diviser la consommation électrique de la phase de build par un facteur significatif par rapport à la moyenne.

4 Autres impacts écologiques et usage de l'IA

4.1 Choix du matériel

Type de machine	Conso. journalière	Conso. annuelle
Portable	0.20 kWh	75 kWh
PC fixe	0.77 kWh	280 kWh

TABLE 2 – Comparaison des consommations matérielles

Le choix du poste de travail constitue un facteur déterminant dans le bilan énergétique global d'un projet logiciel. Les mesures présentées montrent qu'un ordinateur portable consomme en moyenne **près de quatre fois moins d'énergie** qu'un PC fixe standard (0,20 kWh contre 0,77 kWh par jour).

Ce choix a été fait dès le début du projet par l'ensemble de l'équipe. Il permet de réduire non seulement la consommation électrique directe, mais également l'empreinte carbone liée à la fabrication du matériel, les ordinateurs portables ayant en général une durée de vie plus longue et une puissance mieux adaptée à nos usages.

Ainsi, le matériel constitue un premier levier de sobriété numérique, souvent sous-estimé mais pourtant très efficace à l'échelle d'un projet académique.

4.2 Numérique vs déplacements

La comparaison entre l'impact énergétique du projet numérique et celui des déplacements des membres de l'équipe met en évidence un écart très important.

L'empreinte carbone estimée de l'activité informatique (compilation, tests, usage des outils) est d'environ **0,19 kg CO_2 eq**. À titre de comparaison, les déplacements physiques liés au projet (réunions, trajets domicile–école) représentent environ **10,82 kg CO_2 eq**, soit un facteur proche de **57**.

Cette différence montre que, bien que l'optimisation énergétique du logiciel soit nécessaire, l'impact environnemental global d'un projet dépend fortement de facteurs organisationnels et logistiques. Cela souligne l'importance d'une réflexion globale sur la sobriété, dépassant le seul cadre du code.

4.3 Impact de l'IA

4.3.1 Empreinte hydrique

Une conversation avec un modèle d'IA générative peut consommer jusqu'à **1 litre d'eau**, en raison du refroidissement nécessaire des centres de données utilisés pour l'inférence. Ce coût environnemental, bien que souvent invisible pour l'utilisateur, constitue un enjeu réel à l'échelle de l'usage massif des modèles de langage.

Dans le cadre de notre projet, nous avons eu recours à l'IA principalement pour **comprendre les ressources mises à disposition**, en particulier le polycopié (certains concepts techniques abstraits, IMA, processus de compilation, ...) que nous avions du mal à assimiler uniquement à partir du support de cours.

5 Conclusion

Cette analyse met en évidence que l'empreinte énergétique d'un projet logiciel est multifactorielle. Elle ne dépend pas uniquement de la qualité du code généré, mais également de l'organisation du travail, des outils utilisés et des choix matériels.

Grâce aux mesures physiques réalisées (temps CPU, sondes Intel RAPL), nous avons pu quantifier l'impact réel de nos processus. Si l'optimisation de notre chaîne de fabrication a été un succès (temps de build divisés par 50 par rapport à certains groupes), l'analyse intrinsèque via le benchmark *Syracuse* a révélé une **dette énergétique logicielle de 25%** par rapport à l'état de l'art. Ce chiffre souligne que l'optimisation bas niveau (allocation de registres, suppression de sauts inutiles) reste un levier indispensable pour réduire la consommation de l'utilisateur final.

En définitive, cette étude nous a permis de prendre conscience que la sobriété numérique est un enjeu transversal. Le projet génie logiciel illustre comment une démarche responsable, alliant efficacité des processus et mesure de la performance du code, peut être intégrée dès la phase de développement.