Ensimag Grenoble-INP

Compilateur DECA
Analyse des impacts énergétiques

Auteurs :
Oussama Lamzaouri
Mohammed hadi Chamsi
Ziad Razani
Rong Li
Sofia Echarif

Tuteur :
Akram IDANI

27 janvier 2022

Table des matières

1 Quelque généralités sur la consommation énergétique numérique française **2**

2 Consommation énergétique **3**

3 Efficience du code produit : validation des tests **4**
3.1 Pour le projet 4
3.2 Code assembleur généré 4
3.3 Pour l'extension [TRIG] 5

Préambule

Durant ces dernières années, il est de plus en plus clair que nos actions en tant que société se reflètent sur notre environnement, ce qui a un effet direct sur notre quotidien. Ceci nous conduit, en tant qu'ingénieurs responsables, à nous soucier de notre efficacité énergétique et d'essayer d'optimiser notre travail dans le présent pour que nous puissions vivre dans des conditions humaines dans le futur. Ce document présente toutes nos actions pour augmenter notre efficacité énergétique durant le développement de ce projet, des plus impactantes aux plus petits détails qui comptent toujours.

Chapitre 1

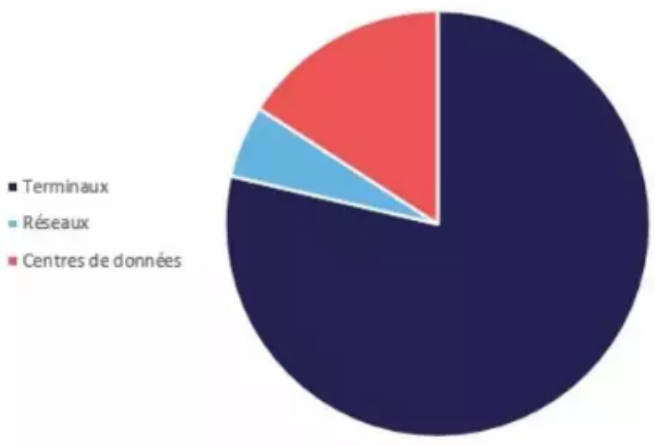
Quelque généralités sur la consommation énergétique numérique française

L'impact des technologies de communications (TIC) sur l'environnement est de plus en plus pointé du doigt. La dématérialisation des services et la complexité des canaux ne doit pas faire oublier qu'envoyer un simple email a des répercussions.

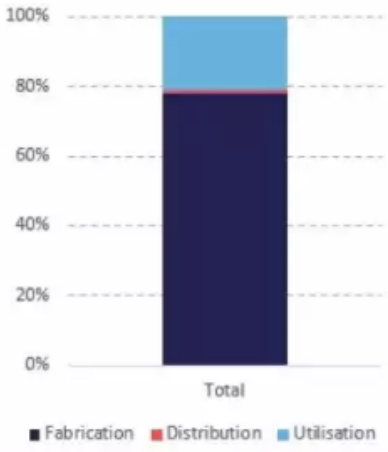
À l'heure où les appareils électroniques sont toujours plus nombreux à gagner notre quotidien, le Ministère de la Transition écologique a souhaité obtenir une évaluation de l'impact des services numériques sur l'environnement en France. L'Arcep et l'Ademe viennent de remettre leurs rapports dans lesquels ils font un état des lieux et émettent un certain nombre de propositions.

10% de la consommation électrique française
Plusieurs points généraux sont mis en avant : il faut donc savoir que les services numériques représentent environ 10% de la consommation électrique française (48,7 TWh pour une consommation de 475 TWh) et sont responsables d'environ 2,5% de l'empreinte carbone de l'Hexagone.

Ils nécessitent 62,5 millions de tonnes de ressources pour produire et utiliser les équipements numériques et génèrent 20 millions de tonnes de déchets par an sur l'ensemble de leur cycle de vie, soit une production de déchets de 299 Kg / habitant.



Part de l'empreinte carbone associée à chaque brique du numérique



Part de l'empreinte carbone associée à chaque phase de l'ensemble des trois briques

Chapitre 2

Consommation énergétique

Pour avoir une image de la consommation énergétique, on s'intéresse premièrement à la composante au cœur de notre système : CPU.

Sous une charge de travail raisonnable, la plupart des processeurs modernes peuvent facilement arriver à 80 Watts voire plus de 100 Watts. Ceci signifie que par cycle, nous avons en moyenne :

$$P_{cycle} = \frac{P_{moyenne}}{Nb_{cycles}} = \frac{100}{3 * 10^9} = 3.33 * 10^{-8} W$$

Cette quantité peut paraitre infime, mais ce qu'elle cache est le grand nombre de cycles CPU gaspillés, ce qui engendre une quantité d'énergie colossale quand on prend en considération une durée comme un mois (Projet GL). Voici un exemple de ce que les CPU/GPU peuvent faire en terme d'énergie : China crypto mining power shortages.

Core	At Boot	Proposed
Reference Clock	100,000 MHz	100,000 MHz
Max Turbo Boost CPU Speed	5,000 GHz	5,000 GHz
Intel® Turbo Boost Technology	Enable	Enable
Turbo Boost Power Max	75,000 W	75,000 W
Turbo Boost Short Power Max	107,000 W	107,000 W
Turbo Boost Short Power Max...	Enable	Enable
Turbo Boost Power Time Win...	28,000 Second	28,000 Second
Core Voltage Mode	Adaptive	Adaptive
Core Voltage	Default	Default
Core Voltage Offset	0,000 V	0,000 V
Processor Core IccMax	140,000 A	140,000 A
AVX Ratio Offset	0,000 x	0,000 x
1 Active Core	50,000 x	50,000 x
2 Active Cores	49,000 x	49,000 x
3 Active Cores	47,000 x	47,000 x
4 Active Cores	46,000 x	46,000 x
5 Active Cores	45,000 x	45,000 x
6 Active Cores	43,000 x	43,000 x
Cache	At Boot	Proposed
Cache Voltage Mode	Adaptive	Adaptive
Cache Voltage	Default	Default
Cache Voltage Offset	0,000 V	0,000 V
Cache IccMax	140,000 A	140,000 A
Other	At Boot	Proposed

FIGURE 2.1 – Spécifications d'un CPU utilisateur : i7-10750H

Chapitre 3

Efficience du code produit : validation des tests

La partie la plus gourmande en ressources est l'étape de validation. L'exécution de l'ensemble des tests prend plusieurs minutes, mais nous travaillons tous sur des branches différentes lorsque nous développons des fonctionnalités en parallèle et seule la fonctionnalité développée est testée sur un nombre limité de tests. L'exécution de tous les tests est exceptionnelle et réservée uniquement à la fusion des branches ou à la vérification de la couverture des tests. La totalité des 311 tests prend 1min10s alors que tester un seul fichier déca prend 0.22s de compilation. Tester uniquement les fonctionnalités qui nous intéressent à ce moment-là est donc beaucoup moins énergivore et plus rapide. Cependant, comme calculé précédemment, la puissance nécessaire pour exécuter les tests n'est que de 5W, l'énergie consommée par les tests est donc négligeable par rapport à la consommation habituelle d'un ordinateur sur une journée.

— **Skip JaCoCo = True** : On évite de lancer JaCoCo à chaque compilation, ceci n'absorbe pas seulement beaucoup plus d'énergie que ce dont on a besoin, mais augmente aussi le temps d'attente.
— **Scripts** : On ne lance les scripts de tests (qui exécutent tous les tests dans une certaine catégorie) qu'à la fin d'une tâche pour tester la non-régression de notre compilateur. Pour déboguer il suffit de lancer des tests seuls.
— **Utilisation de mvn test** : Comme pour JaCoCo, l'utilisation de mvn test se fait après chaque tâche et non au plein milieu d'un travail de débogage. À la fin de cette dernière, l'exécution de cette commande offre un diagnostic qui permet de soit déboguer des tests unitaires non réussis ou de passer à la tâche suivante.

3.1 Pour le projet

— '**Privilégier le travail à l'école** : Les locaux étant déjà ouverts et alimentés pour d'autres élèves, il est plus judicieux (et eco-friendly) d'y travailler par équipe que d'occuper 5 différents emplacements, chacun alimenté par sa propre source d'énergie.

3.2 Code assembleur généré

Au tout début, le code généré manquait d'optimisation. Par exemple, puisqu'il nous semblait impossible de prédire le nombre de variable globales et de méthodes qu'allait contenir un programme, on s'est senti obligé de tester, à chaque fois qu'on voulait réserver plus d'espace dans la pile pour une nouvelle variable ou une nouvelle méthode, si la pile dispose d'espace. Au final, on a découvert qu'il ne fallait pas prédire le nombre de variables pour contourner ce problème, et en résolvant celui-ci, on a pu économiser du temps et de l'énergie. (Le code assembleur qu'on génére n'est pas vraiment du code assembleur, c'est un programme qui sera interprété, on discutera néanmois, la consommation d'un code assembleur directement exécutable par une machine). Chaque instruction dans un langage assembleur est un accès mémoire, chaque accès mémoire est susceptible de provoquer le changement d'état d'au moins un transistor, et l'érergie epuisée par chaque changement d'état de transistors CMOS peut être quantifiée :

$$P_{avg} = a \cdot C_L \cdot V_{DD}^2 \cdot f_{\texttt{clk}}$$

C_L, étant la capacité totale ayant changé d'état, V_{DD}, la tension d'alimentation, et $f_{\texttt{clk}}$ étant la fréquence d'horloge et α étant un réel (Node transition activity factor).

Nous observons, qu'en effet, plus l'on élemine d'insctructions dans le code qu'on génére, plus l'on économise de l'énergie et de temps.

3.3 Pour l'extension [TRIG]

L'extension [TRIG] fournit des fonctions indispensables à tout programmeur, ceci nous pousse à améliorer davantage ses performances puisqu'ils seront au cœur de la plupart des programmes complexes et se répèteront à chaque fois tout en consommant de l'énergie. Ceci a aussi pour bénéfice d'améliorer les performances non seulement énergétiques, mais aussi au niveau complexité.

— **Coefficients calculables** : Au lieu de recalculer les premiers coefficients de Taylor à chaque appel d'une fonction trigonométrique, il est bien plus efficace de les avoir écrits en dur pour éviter même d'avoir à effectuer une division, mis à part l'opération récursive de calcul des nombres factoriels.

— **Étude de la limite de précision** : En effet il y a une limite de précision utile qu'on pourrait avoir pour notre programme en dessous de laquelle l'aspect machine ne pourra pas distinguer entre deux valeurs. Il est alors inutile de "rallonger" les séries pour approximer les fonctions lorsque la convergence est déjà rapide.

Bibliographie

[1] Energy complexity of software in embedded systems.
`https://arxiv.org/ftp/nlin/papers/0505/0505007.pdf`.

[2] Christan D. Impact des services numériques sur l'environnement.
https://www.insis.cnrs.fr/fr/cnrsinfo/reduire-limpact-environnemental-du-calcul-..., 2022. [Online ; accessed 20-Janvier-2022].