le cnam

Alexandra SEMEUR
alexandra.semeur.auditeur@lecnam.net

Alexandre TORTEVOIS
alexandre.tortevois.auditeur@lecnam.net

Green Computing

SMB215 : Infrastructure technologique et confiance
Suivi par Pierre Paradinas

Table des matières

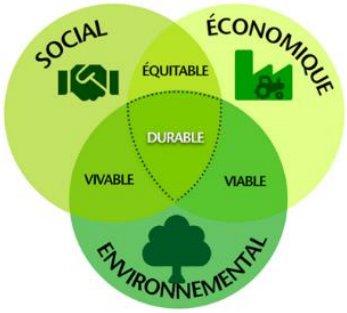
I. Introduction

Les expressions « développement durable », « écoconception » ou encore « transition écologique », sont désormais couramment employées. En effet, la prise de conscience à l'égard des problèmes environnementaux s'est amplifiée ces dernières années, tant au niveau politique que citoyen, notamment à cause de la multiplication du nombre de catastrophes naturelles liées au dérèglement climatique. A ce titre, il y a eu, par exemple, les Accords de Paris sur le climat lors de la COP21 en 2015 [1], l'organisation de la première marche pour le climat en septembre 2018 en France, la Conférence de Madrid sur les changements climatiques lors de la COP25 en 2019 et bien d'autres événements.

Or, ces notions ne sont pas nouvelles mais sont apparues au début des années 1970. En effet, dès 1972, la Conférence des Nations Unies sur l'environnement qui s'est tenue à Stockholm a été la première conférence mondiale qui a fait de l'environnement une question majeure. Les participants y ont adopté une série de principes pour une gestion écologiquement rationnelle de l'environnement. Puis en 1987, le rapport Brundtland, rédigé par la Commission mondiale sur l'environnement et le développement de l'ONU, a définie pour la première fois la notion de développement durable comme suit :

> « Le développement durable est un développement qui répond aux besoins du présent sans compromettre la capacité des générations futures de répondre aux leurs. Deux concepts sont inhérents à cette notion :
> - Le concept de « besoins », et plus particulièrement des besoins essentiels des plus démunis, à qui il convient d'accorder la plus grande priorité, et
> - L'idée des limitations que l'état de nos techniques et de notre organisation sociale imposent sur la capacité de l'environnement à répondre aux besoins actuels et à venir. »

A la suite de ce rapport, en 1988, l'ONU fonde le panel inter-gouvernemental sur les changements climatiques (IPCC) pour évaluer les changements climatiques.



Le Green Computing s'inscrit dans une démarche visant à réduire l'empreinte écologique, économique et sociale des Technologies de l'Information et de la Communication (TIC) tout au long du cycle de vie d'un produit, de sa conception à son recyclage. En d'autres termes, c'est faire une utilisation plus respectueuse de l'environnement et des ressources dans les systèmes informatiques.

Les objectifs sont donc :

- Réduire l'empreinte environnementale des systèmes informatiques
- Permettre aux acteurs de réaliser des économies
- Conserver la simplicité d'accès et d'utilisation

Dans cette étude nous nous intéresseront aux aspects tant réglementaires, matériels que logiciels qui rendent les systèmes informatiques plus « green ».

II. Histoire du Green Computing

L'histoire du Green Computing est jalonnée de différentes certifications, écolabels et normes apparus depuis 1977 et, plus récemment, de composantes législatives européennes et françaises.

Cette section recense les différents écolabels et s'attarde sur les plus reconnus et usités. Nous nous intéresserons également aux exigences des très récents aspects législatifs et à leurs impacts dans les prochaines années.

A. Les écolabels

1. Synthèse

Le tableau en annexe A liste les écolabels existants [2] et applicables au domaine de l'IT.

La plupart des écolabels considèrent l'ensemble du cycle de vie ou seulement la consommation énergétique d'un produit et sont attribués par des organismes indépendants.

On notera que certaines grandes entreprises comme Fujitsu-Siemens, P&G et Philips ont créé leur propre label pour leurs produits.

Parmi ces 30 écolabels, 5 sont plus largement reconnus et bénéficient de la confiance des entreprises, des gouvernements et des consommateurs.

Le GEN (Global Ecolabelling Network), association à but non-lucratif créée en 1994, a pour but de fédérer les labels autour de critères communs (dont l'obtention de la norme ISO14024).

2. Blue Angel (1978)

Label créé en 1978 en Allemagne [3] et décerné par un jury composé de 13 membres. Les produits concernés doivent à la fois tenter de réduire les effets de leur production sur l'environnement tout en étant de même qualité, fiabilité et sécurité que les autres. Plus de 4000 produits l'ont obtenu.

3. Energy Star (1992)

Crée en 1992, ce programme soutenu par l'U.S. Environmental Protection Agency (EPA), est reconnu comme une référence présentant l'efficacité énergétique d'un produit. Son objectif est de protéger le climat et la santé des personnes en réduisant les émissions de gaz à effet de serre et les déchets. Ce label n'est pas destiné uniquement aux équipements informatiques et s'obtient après des tests réalisés par des laboratoires indépendants et reconnu par l'EPA. C'est actuellement la version 8.0 d'Avril 2020 qui est utilisée pour définir les critères d'éligibilités des équipements informatiques [4].

4. TCO (1992)

Première certification mondiale de durabilité pour les produits informatiques, TCO répond aux exigences définies par la norme ISO 14024 et fait partie du Global Ecolabelling Network. L'obtention du label requiert des tests réalisés par des organismes indépendants respectant la norme ISO17025. [5]

5. 80 Plus (2004)

Initialement, ce label visait à promouvoir l'utilisation d'alimentations électriques plus efficientes dans les ordinateurs de bureau. Depuis, il a évolué en 'Ecos Plug Load Solutions program' qui promeut l'utilisation d'une large gamme de technologies moins consommatrices d'énergie. [6]

6. EPEAT (2006) – IEEE 1680

Le sigle EPEAT signifie : "Electronic Product Environmental Assessment Tool". Il est basé sur le standard IEEE 1680 (Standard for Environmental Assessment of Electronic Products) [7] et est géré par le Global Electronics Council (GEC), une organisation à but non lucratif dédiée à la création d'un monde plus juste et durable. Il vise à évaluer l'impact environnemental des équipements électroniques en fonction de 51 critères (23 obligatoires et 28 optionnels) couvrant une grande variété de domaines (cycle de vie complet du produit, choix des composants, longévité du produit, économie d'énergie, empreinte carbone, …). Les produits sont classés suivant 3 niveaux :

Bronze	Silver	Gold
Tous les critères obligatoires	Tous les critères obligatoires et au moins 50% des critères facultatifs	Tous les critères obligatoires et au moins 75% des critères facultatifs

B. Les normes

1. ISO26000 : Responsabilité Sociétale (2010)

La norme ISO26000, est, comme toutes les normes ISO, définie et régie par l'organisation internationale de normalisation (en anglais International Organization for Standardization). Cette norme établie les lignes directrices relatives à la responsabilité sociétale des entreprises [8], c'est à dire leur manière de contribuer au développement durable. Cette norme, qui ne donne pas lieu à une certification, centralise son approche autour de sept problématiques :

- La gouvernance de l'organisation
- Les droits de l'homme
- Les relations et conditions de travail
- L'environnement
- La loyauté des pratiques
- Les questions relatives aux consommateurs
- Les communautés et le développement local

2. ISO14001 : Système de Management Environnemental (2015)

La norme ISO140001 [9] spécifie les exigences d'éco-management. Elle met l'accent sur la sensibilisation du personnel et le traitement des demandes externes et, à moindre échelle, la communication externe. Elle comporte 18 exigences réparties en 6 chapitres. Les exigences générales sont :

- La politique environnementale
- La planification
- La mise en œuvre des actions pour satisfaire à la politique environnementale
- Les contrôles et les actions correctives
- La revue de la direction

3. ISO50001 : Système de Management de l'Énergie (2018)

La norme ISO50001 [10] a été élaborée par 61 pays en 2011 et vise à l'amélioration de la performance énergétique. En se basant sur un diagnostic énergétique initiale, l'entreprise définie ses cibles énergétiques et met en place un plan de comptage de l'énergie. Ce système de management permet de réaliser des économies d'énergie et de réduire les coûts.

C. Aspect législatif

1. Européen

a. Directive européenne RoHS (2003)

Le sigle RoHS signifie : "Restriction of Hazardous Substances" (in electrical and electronic equipment). L'objectif de cette directive est de limiter l'utilisation de substances dangereuses dans les équipements électriques et électroniques. Les substances concernées sont le plomb, le mercure, le chrome hexavalent, les retardateurs de flamme polybromobiphényles (PBB) et polybromodiphényléthers (PBDE) dans la limite de 0.1% par unité de poids et le cadmium dans la limite de 0.01%.
L'amendement 2015/863 [11], entré en vigueur le 22 juillet 2019 ajoute à cette liste 4 nouvelles substances : Le phtalate de bis-(2-éthylhexyle) (DEHP), le phtalate de benzyle et de butyle (BBP), le phtalate de dibutyle (DBP) et le phtalate de diisobutyle (DIBP) dans la limite de 0.1% par unité de poids. Les produits commercialisés en Europe doivent répondre à cette directive.

b. Label 2020 : Étiquette énergétique

L'Union européenne a voté la mise en place d'une évolution des étiquettes énergétiques à partir de 2021. Les fabricants ont désormais l'obligation d'enregistrer leurs produits dans une base de données européenne (EPREL) [12]. Cette nouvelle étiquette se veut plus efficace et plus lisible, pour permettre aux consommateurs d'identifier clairement les produits les plus économes en énergie et de contribuer à une économie plus verte. Elle concerne notamment les écrans d'ordinateurs.

c. Green Deal

L'European Green Deal [13] a pour but de rendre l'Europe climatiquement neutre à l'horizon 2050. Cet ensemble d'initiatives élaborées par la commission européenne prévoit, entre autres, l'investissement dans les technologies respectueuses de l'environnement.

2. Français

a. Loi Grenelle 2

La loi Grenelle 2 [14], qui complète et met en application la loi Grenelle 1, a été promulguée le 12 juillet 2010. Elle impose l'établissement d'un bilan des émissions de gaz à effet de serre aux entreprises de plus de 500 salariés en métropole et de plus de 250 salariés en outre-mer ainsi que la mise en place d'un plan climat-énergie territorial aux établissements publics de plus de 250 personnes et aux collectivités territoriales de plus de 50 000 habitants.

b. Loi Anti-gaspillage : Indice de réparabilité

La loi anti-gaspillage du 10 février 2020 a pour but de promouvoir l'économie circulaire, c'est-à-dire le fait de réutiliser, réparer, recycler, et produire de manière éco-responsable. Elle prévoit, entre autres, l'affichage systématique de l'indice de réparabilité d'un produit électronique [15].

c. Loi Climat & Résilience

Trois articles de la loi Climat et résilience (toujours à l'étude par le Sénat et l'Assemblée nationale), proposent d'ajouter l'écoconception des logiciels et service numériques aux cursus des ingénieurs et techniciens de développement logiciel [16].

III. Comment rendre l'informatique durable ?

Le fonctionnement des système informatiques, notamment dans le cadre des datacenters, représente une très grosse partie de leur impact environnemental. Dans cette partie nous aborderons différents moyens et techniques qu'il est possible de mettre en œuvre pour réduire celui-ci.

A. La Gestion des ressources

1. Utilisation d'énergie verte

Le Green Computing favorise l'utilisation d'énergie verte telle que l'énergie éolienne, solaire, géothermique, etc...
On peut citer quelques projets d'énergie verte dans le domaine IT:

- Projet Formosa : un parc éolien offshore au large des côtes de Taïwan [17]
- Google finance 18 projets d'énergie éolienne et solaire en Europe, aux USA et au Chili pour un total de 1.6GW [18]
- Ørsted a signé un accord d'achat d'énergie renouvelable pour fournir 920 MW d'énergie éolienne offshore au fabricant taïwanais de semi-conducteurs TSMC, un fournisseur d'Apple [19]
- OVH a construit et déployé ses propres éoliennes [20]
- Google a pour objectif de ne plus émettre de CO2 en 2030 [21] et choisi avec soin l'emplacement géographique de ses datacenters afin de limiter leur consommation et optimiser la production d'énergie. [22]
- Amazon a pour but d'alimenter ses activités avec 100% d'énergie renouvelable d'ici 2025 et à un objectif de zéro émission de carbone d'ici 2040 [23]

2. Réduction de la consommation d'eau

L'IT consomme de l'eau pendant le processus de fabrication des composants et pour le refroidissement des équipements.

Le processus de fabrication des puces électroniques consomme de grandes quantités d'eau. La fabrication d'une puce électronique de 1cm² demande par exemple de 18 à 27L d'eau [24]

Concernant le refroidissement des équipements et des datacenters notamment, des projets voient le jour pour tenter de mettre en place des solutions de refroidissement plus efficientes.

On citera par exemple le projet Natick qui consiste à déployer des datacenters sous-marins refroidis par l'eau de mer (et utilisant des énergies marines). [25] , un autre projet de Microsoft en coopération avec 3M consistant à plonger les serveurs dans un liquide non-conducteur porté à ébullition à 50°C. [26] ou encore Scaleway qui a pour objectif de faire baisser la consommation d'eau de ses datacenters à 0.15L/kWh contre 1.8L/kWh pour la plupart des datacenters aujourd'hui [27]

3. Sensibilisation aux écogestes

Le Green Computing passe également par le changement des habitudes d'utilisation des appareils électroniques. Les points primordiaux sont :

- L'arrêt plutôt que la mise en veille
- L'utilisation de réseaux de communication filaires ou Wifi plutôt que 3G/4G/5G
- La limitation des flux de données (éteindre la webcam lors des visioconférences, stocker les données en local plutôt qu'en ligne, désactiver la synchronisation automatique, désactiver la lecture automatique, etc.…)
- La limitation de la taille des flux en redimensionnement les images, compressant les fichiers, etc...
- L'allongement de la durée de vie des appareils en installant, par exemple, un système d'exploitation dédié au matériel "obsolète" comme LineAgeOS pour les smartphones Android.

B. Le Matériel

Les principales sources de consommation d'énergie sont le fonctionnement et le refroidissement

Pour réduire l'énergie consommée lors du fonctionnement des appareils, il convient de sélectionner les bons composants lors de l'achat :

- Processeur : rechercher la plus grande finesse de gravure (par effet de cascade, un processeur consommant 1W de moins entrainera une consommation réduite de 2.84W au niveau de son datacenter) [28]. Entre 2018 et 2020, la finesse de gravure est passée de 7nm à 5nm entrainant un gain de consommation de 21%. [29] (voir annexe B)
- Stockage : les disque durs SSD consomment moins d'énergie qu'un HDD quelle que soit la tâche effectuée [30]
- Carte graphique : limiter la puissance à l'utilisation ciblée
- Alimentation : opter pour des alimentations intelligentes qui adaptent la tension à la demande
- Moniteur : privilégier le LCD et limiter la taille

C. La Virtualisation

La virtualisation permet de regrouper plusieurs machines virtuelles sur une même machine physique. Même si le matériel est d'avantage sollicité, les gains en matière de consommation d'énergie sont réels. L'utilisation d'orchestrateurs permettant de gérer les ressources et la mise en fonctionnement/arrêt des VM améliore ce gain.

Certaines études avancent que la virtualisation entrainerait un gain de consommation de 8% pour un datacenter de 465m² [31].

Les nouvelles solutions de conteneurisation comme Docker peuvent apporter un gain supplémentaire si elles sont couplées à des solutions d'orchestration comme Kubernetes [32] mais nativement, elles augmentent la consommation de 2 kW en moyenne [33].

Le projet d'Internet 2.0, Dfinity, que nous avons étudié en préparant notre exposé « Blockchain », est basée sur l'utilisation de conteneurs pour l'ensemble des applications web et pourrait constituer un vecteur de réduction de la consommation d'énergie de l'IT s'il devenait massivement utilisé.

D. Le monitoring

Le monitoring consiste à surveiller et mesurer la consommation de ressources tant au niveau logiciel que matériel. Cette surveillance peut se baser sur des capteurs internes (embarqués dans les composants) et/ou des capteurs externes (sondes de température, sonde d'hygrométrie, capteur de consommation électrique). Ces données sont ensuite utilisées afin d'optimiser les ressources.

La thèse de Hayri Acar de mars 2018 [34], liste l'ensemble des outils de monitoring de composants existants. Ceux-ci sont classés en fonction du composant qu'ils monitorent et de leur état d'avancement.

Solution proposal	Solution pour un problème particulier
Validation Research	Technique implémentée, évaluation en cours
Evaluation Research	Technique nouvelle non-encore implémentée

Ces outils sont tous plus ou moins spécifiques à un environnement ou un composant en particulier et nombre d'entre eux reposent sur l'utilisation de commandes logicielles fournies par les OS. On citera notamment PowerAPI, un des outils les plus aboutis, qui permet d'estimer l'énergie consommée par un processus en temps réel mais qui est cependant limité aux architectures Intel et orienté Docker

Pour notre expérimentation, nous avons donc choisi d'utiliser ces commandes natives directement.

E. L'écoconception

L'écoconception est la prise en compte de l'impact environnemental dès la conception du produit.

Au niveau du développement logiciel, cette approche n'en est qu'à ses balbutiements mais certains acteurs majeurs comme Microsoft tente de mettre en place des principes [35]. D'autres modèles, comme GREENSOFT par exemple, propose également une méthodologie dans laquelle les pratiques Green IT sont utilisées, ce qui permettra de réduire la consommation d'énergie des ordinateurs tout en développant des logiciels, s'ils sont mis en œuvre [36].

La loi Climat & Résilience de 2021 souhaite introduire l'écoconception logicielle dans les cursus de formation des développeurs informatiques.

IV. Travail d'expérimentation

Pour notre expérimentation, nous nous sommes intéressés à comment mesurer la quantité d'énergie consommée par l'exécution d'un programme et comment en améliorer son efficience. Nous avons donc comparé l'exécution d'un même programme traduit en plusieurs langages.

A. État de l'art

En 2017, des chercheurs de l'université de Minho au Portugal [37] ont réalisé une étude comparative entre 27 langages de programmation en utilisant 10 algorithmes de benchmark dont les sources sont disponibles sur GitHub [38] et dont les résultats ont été présentés lors de la conférence SLE'17 (Software Language Engineering des 23/24 Octobre 2017).

Lors des exécutions les paramètres suivants ont été mesurés :

- le temps d'exécution
- l'utilisation de la mémoire
- l'énergie consommée

Les langages ont été classé suivant un diagramme de Pareto selon une combinaison des trois critères précédents :

Time & Memory	Energy & Time	Energy & Memory	Energy & Time & Memory
C • Pascal • Go	C	C • Pascal	C • Pascal • Go
Rust • C++ • Fortran	Rust	Rust • C++ • Fortran • Go	Rust • C++ • Fortran
Ada	C++	Ada	Ada
Java • Chapel • Lisp • Ocaml	Ada	Java • Chapel • Lisp	Java • Chapel • Lisp • Ocaml
Haskell • C#	Java	OCaml • Swift • Haskell	Swift • Haskell • C#
Swift • PHP	Pascal • Chapel	C# • PHP	Dart • F# • Racket • Hack • PHP
F# • Racket • Hack • Python	Lisp • Ocaml • Go	Dart • F# • Racket • Hack • Python	JavaScript • Ruby • Python
JavaScript • Ruby	Fortran • Haskell • C#	JavaScript • Ruby	TypeScript • Erlang
Dart • TypeScript • Erlang	Swift	TypeScript	Lua • JRuby • Perl
JRuby • Perl	Dart • F#	Erlang • Lua • Perl	
Lua	JavaScript	JRuby	
	Racket		
	TypeScript • Hack		
	PHP		
	Erlang		
	Lua • JRuby		
	Ruby		

Leur constat est qu'un langage moins énergivore est le résultat d'un compromis entre la rapidité d'exécution et la quantité de mémoire consommée.

Le raccourci « plus un langage est rapide, plus il est efficient » n'est ainsi pas toujours vrai. Ainsi, par exemple, le Pascal et le Go sont plus énergivores mais consomment moins de quantité de mémoire ce qui les rend globalement plus efficients, d'après les résultats et conclusions de cette étude.

B. Mise en œuvre

Pour la mise en œuvre de la partie expérimentale, nous avons choisi de résoudre un problème proche de ce qui pourrait être posé et résolu en entreprise. Nous nous sommes basés sur une version simplifiée d'un problème posé lors du Google Hash Code 2014 disponible sur le site https://primers.xyz/0

1. Outils et moyens de mesure employés

L'expérimentation a été réalisée sur un nano-ordinateur Raspberry Pi 3B+ (voir annexe C) et son système d'exploitation dédié (Raspberry Pi OS Light). 5 langages de programmation différents ont été utilisé, dont 4 sont très largement utilisés dans les applications Web :


C (référentiel)


Python


Java

PHP

JavaScript

Pour réaliser les mesures nous avons utilisé un capteur de courant (TI INA260) [39]. Celui-ci a été instrumenté sur un second Raspberry Pi 3B+ dont l'unique but est de servir de centrale d'acquisition. Le schéma de montage est fourni en annexe D.

Les paramètres d'acquisition suivant ont été choisi :

- Averaging Mode à 4, ce qui signifie que la valeur lue sera moyennée sur 4 points.
- Conversion Time à 588µs, ce qui signifie que le temps d'acquisition d'un point de mesure dure 588µs

Avec cette configuration, le capteur a la capacité de nous fournir une valeur de courant et de tension toutes les 5ms (2 (U&I) x 4 point x 0.588ms = 4.704ms).

La validité des mesures réalisées par le capteur a été vérifiée préalablement à la mise en place de l'expérimentation (voir annexe E)

Les commandes natives perf [40] et time ont été utilisées pour analyser les performances du CPU et mesurer la quantité de mémoire utilisée par le processus.

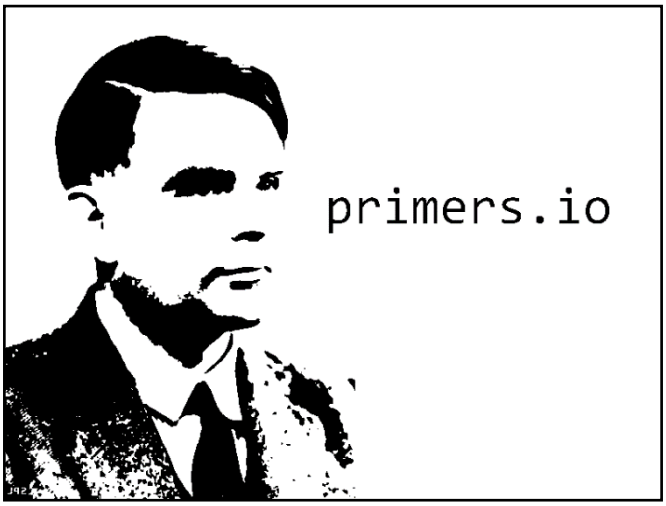
De nombreuses ressources nous ont aidé à comprendre les résultats obtenus avec la commande perf et nous ont conforté dans notre choix d'utiliser cette commande pour réaliser nos analyses [41] [42] [43].

L'exécution des programmes étant mono-thread, pour s'affranchir des migrations liées à l'ordonnanceur qui pourraient fausser nos résultats, nous avons bloqué nos exécutions sur un seul cœur du processeur à l'aide de la commande taskset.

Un script en Python a été écrit pour automatiser l'exécution des différents programmes et en récupérer les résultats. L'ensemble des sources qui ont été écrites et utilisées pour cette expérimentation est disponible sur GitHub et le détail des algorithmes est fourni en annexes du présent rapport.

2. Algorithmes

L'objectif du problème est de générer un nombre minimum d'instructions pour « peindre » une image composée de pixels noirs et blancs :



Nous disposons pour cela :

- d'un fichier texte en entrée, dont les caractères présents sur chaque ligne représentent l'état de chaque pixel de l'image (* = vide, # = peint)
- des instructions :
  - FILL,x,y,size : peint un carré de taille size dont le côté en haut et à gauche est en x, y
  - ERASE,x,y : efface la peinture du pixel en x, y

Pour simplifier la résolution du problème, nous avons écarté la possibilité de réaliser des effacements.

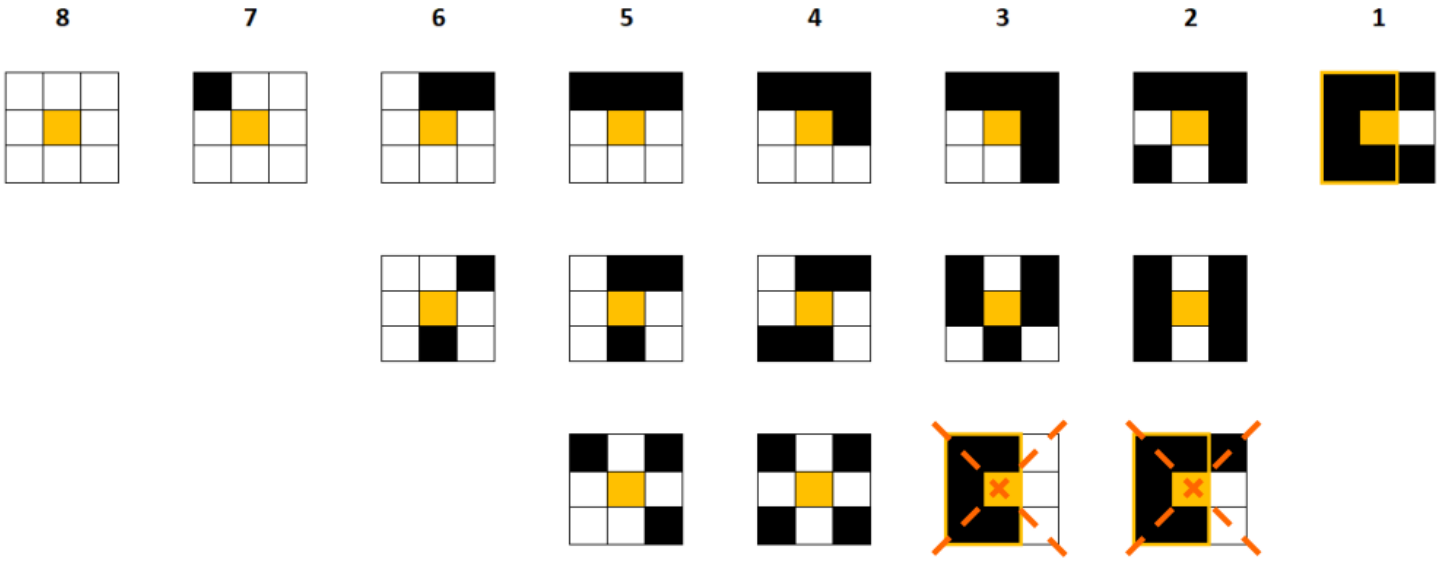
La taille de l'image est de 800x600 pixels, soit un total de 480 000 pixels.

Une première solution (algo-01), très basique, consiste, pour chaque pixel à peindre, à générer une instruction (un carré de 1x1). On obtient un résultat de 84 784 instructions, c'est le nombre de pixels peints sur l'image. (voir annexe F.1)

Une seconde solution (algo-02) est de remplir avec le plus grand carré disponible à partir de chaque pixel. On effectue donc pour les 84 784 pixels à peindre un pré-calcul de la plus grande zone qu'il est possible de peindre à partir de celui-ci. On parcourt ensuite l'image et, dès qu'un pixel est « à peindre », on remplit avec la plus grande zone possible. On obtient un résultat de 3 146 instructions (voir annexe F.2).

Une troisième solution (algo-03) consiste à calculer un indice d'isolement pour chaque pixel, correspondant au nombre de pixels vides en périphérie immédiate. Plus un pixel sera isolé et moins il y aura de possibilité pour le remplir (voir annexe F.3).

Voici différents exemples de calcul de l'indice d'isolement :



A partir de l'indice 3, il faut en plus regarder la position des pixels vides pour éliminer les pixels vides non pertinents, car le pixel cible peut s'inscrire dans un carré plus grand. Ainsi pour les index 2 et 3 on élimine le résultat du calcul si les pixels vides sont alignés.

Combiné au pré-calcul précédent on remplit l'image par les pixels les plus isolés, avec le carré le plus efficient disponible jusqu'à l'indice 2 puis on finalise le remplissage en partant de la dernière ligne.

On obtient ainsi un résultat de 3 102 instructions.

3. Exploitation des résultats

Quelques explications préliminaires sont nécessaires pour comprendre l'exploitation des résultats qui est présentée ci-après :

- Chaque exécution, dans chaque langage, a été réalisé consécutivement 10 fois de suite et les résultats ont été moyennés.
- L'implémentation de chaque algorithme a été faite de la façon la plus similaire possible dans chaque langage, en fonction des possibilités d'implémentation donnée dans chaque langage et sans avoir recours à des librairies tierces.
- La consommation, le temps d'exécution et la quantité de mémoire utilisée ont été uniformisé par rapport à la référence en C. Les ratios représentent donc un coefficient multiplicateur par rapport à la référence.
- Le nombre d'instructions par cycle (IPC), le pourcentage de défaut de branche et le pourcentage de défaut de cache sont les données brutes issues du résultat de la commande *`perf`*
- IPC : Instruction per Cycle => Nombre d'instructions exécutées / Nombre de cycles effectués par le CPU
- Défaut de Branche => Nombre de branchements ratés / Nombre de branchements total * 100
- Défaut de Cache => Nombre d'accès au cache ratés / Nombre d'accès au cache total * 100

Commentaires sur la synthèse des résultats des exécutions de l'algo-01 (voir annexe G.1) :

On constate à ce stade, sur cet exemple, que :

- Java et Javascript sont les plus gourmands en mémoire
- Java est le plus énergivore

L'impact de la quantité de mémoire utilisée sur la consommation se visualise quand on compare les ratios de temps d'exécution avec les ratios de consommation. Par exemple, pour PHP les ratios sont respectivement de 3,33 et 3,30 alors qu'en Javascript on a 5,93 et 6,60 et en Java 14,96 et 17,41. Cette augmentation du ratio de consommation est le reflet de la grande quantité de mémoire consommée par ces langages.

Commentaires sur la synthèse des résultats des exécutions de l'algo-02 (voir annexe G.2) :

Après le premier batch d'exécution, nous nous sommes rendu compte que la première implémentation en C ("origin" dans le tableau) était très loin d'être optimale. En effet le nombre de branchement et d'accès au cache étaient bien supérieur par rapport au Java ou Javascript , le temps total d'exécution s'en retrouvant très fortement pénalisé.

Nous avons donc repris l'implémentation en C en utilisant une liste chainée ("linkedlist" dans le tableau) des 84 784 pixels à peindre au lieu de parcourir un tableau de 480 000 pixels à chaque tour de recherche. Le résultat est sans appel : le nombre d'instructions total exécutées et le nombre d'accès au cache sont très fortement diminués.

Néanmoins on observe pour tous les langages un nombre de défaut d'accès au cache très important. Cela signifie que notre système passe beaucoup de temps à charger de la mémoire. L'implémentation en état n'est donc pas des plus optimales en termes de performances.

Commentaires synthèse des résultats des exécutions de l'algo-03 (voir annexe G.3) :

On constate ici que bien que nous ayons complexifié un peu le pré-calcul, les résultats restent très proches de l'algo-02. Néanmoins, pour le Javascript, on observe une dégradation des performances très marquée avec un temps d'exécution augmenté de 11 secondes (+30%) et, bien que le temps d'exécution soit plus court que l'implémentation en C "origin", la performance énergétique globale est moins bonne du fait de l'utilisation d'une plus grande quantité de mémoire et d'un pourcentage de défaut de branche plus important.

4. Pistes d'améliorations

Dans le cadre de ce travail d'expérimentation, après analyse de nos codes sources, nous avons eu quelques idées pour améliorer l'efficience de l'exécution de notre algorithme :

1. Effectuer un pré-calcul pour chaque ligne et chaque colonne de la quantité de pixels à peindre. On évite de parcourir ainsi les lignes/colonnes sans pixel à peindre. On prendra soin de soustraire les pixels à chaque fois qu'on remplit une zone pour éliminer les lignes/colonnes inutiles.
2. Regrouper les pixels à peindre par index d'isolement. Cela évitera de reparcourir 7 fois l'ensemble des pixels pour trouver ceux qui répondent au critère d'isolement pour cibler le remplissage.
3. Lors de la recherche de la zone la plus optimale à peindre pour un pixel ciblé, au lieu de reparcourir tous les pixels de l'images, on pourrait se concentrer sur une zone plus restreinte en ayant conservé la taille du plus grand carré possible. Pareillement, une fois les coordonnées X, Y dépassées ce n'est pas la peine de continuer la recherche. Dans cet exemple, la taille maximum est 82, soit une recherche limitée à 6 724 et non 84 784.
4. Effectuer une recherche « en arrière » de la plus grande zone, pour ne pas faire 6 724 si on trouve des pixels vides qui rendent inutiles la recherche d'un carré plus grand.

L'implémentation des améliorations a été réalisé en Javascript :

- la piste 1 allonge le délais d'exécution de 44 sec
- la piste 2 allonge le délais d'exécution de 6 sec
- la piste 3 améliore le délais d'exécution de 38 sec. ; celle-ci a été implémenté dans tous les langages et est commentée ci-après
- la piste 4 améliore encore le délais d'exécution de 1.4 sec. (algo-03-r2, implémentation disponible sur GitHub)

Commentaires sur la synthèse des résultats des exécutions de l'algo-03-r1 (voir annexe G.4) :

On constate que pour l'ensemble des langages le pourcentage de défaut d'accès au cache chute. Quand on regarde les résultats bruts (disponible sur GitHub), on s'aperçoit que le nombre total d'instructions, de branchements et de chargements de cache sont moins importants, ce qui explique cette nette amélioration des performances.

La piste 4 a été développé uniquement en Javascript, mais les constats finaux sont les mêmes qu'avec la piste 3 : il y a moins d'instructions exécutées, moins de branchements et moins de chargements de cache ce qui permet d'améliorer encore les performances.

5. Synthèse de l'expérimentation

Cette expérimentation nous aura permis de trouver des résultats cohérents par rapport à l'étude de l'université de Minho :

- Un langage compilé sera plus performant, à condition que l'implémentation soit également performante
- La quantité de mémoire utilisée par un programme et les principes de localité spatiale et temporelle influent sur la consommation globale du programme

Nous retiendrons que pour rendre un programme efficient, un développeur doit :

- réfléchir à l'implémentation de son code et garder un esprit critique sur celle-ci
- refactoriser son code pour corriger les performances : l'implémentation la plus simple ou la plus évidente, n'est pas forcément la plus performante ! Il faut éviter les accès mémoires et les branchements (tests conditionnels) inutiles

- faire du profiling pour analyser les performances au travers de plusieurs essais pour trouver où le programme « perds » du temps. Une modification du code peut avoir un impact important (en positif comme en négatif). Les optimisations doivent être réalisées itérativement.
- passer du temps qui peut paraître « improductif » à proprement parler, sauf si l'objectif est de gagner en consommation énergétique

V. Conclusion

Les conclusions que nous pouvons tirer de cette étude sont multiples.

Tout d'abord, au niveau régulation, on constate une grande disparité entre les écolabels. Même si le Global Ecolabelling Network tente de fédérer les écolabels autour de critères communs internationalement reconnus, les critères secondaires quant à eux restent très disparates et limiter à certaines régions uniquement. Les normes ISO ont des critères plus stricts mais elles ne constituent pas non plus une obligation, les entreprises n'étant pas légalement contraintes à l'obtention de celles-ci pour mettre un produit sur le marché. Des lois se mettent en place progressivement, souvent basées sur un affichage obligatoire des performances énergétiques, mais la décision de sanctionner des produits moins respectueux de l'environnement revient alors au consommateur.

Dans un souci tant marketing qu'économique, certains grands acteurs du monde de l'informatique développent des solutions afin de réduire leur empreinte écologique en mettant sur le marché des produits moins consommateurs d'énergie, en réduisant leur consommation de ressources durant les processus de fabrication et en rationalisant les ressources durant la phase d'exploitation.

Au niveau logiciel, l'éco-conception est, pour le moment, basée sur le bon sens d'avantage que sur de réelles règles vérifiables et quantifiables. L'optimisation du code requiert l'expertise du développeur dans le langage adéquat mais aussi un investissement en temps afin de monitorer l'exécution du code pour en comprendre les processus consommateurs de ressources et les améliorer.

Le Green Computing semble donc avoir encore de nombreuses voies d'améliorations possibles. L'impact économique, la prise de conscience collective et les différentes réglementations qui voient le jour devraient permettre l'essor de solutions plus respectueuses de l'environnement d'ci une vingtaine d'années.

VI. Bibliographie

1. *The Paris Agreement.* [En ligne] https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement.

2. Index Ecolabel. [En ligne] http://www.ecolabelindex.com/ecolabels/?st=country,ca.

3. Ecolabel, Blue Angel | The German. Blue Angel | The German Ecolabel. [En ligne] https://www.blauer-engel.de/en.

4. Energy Star Product Specification for Computers v8.0. [En ligne] https://www.energystar.gov/sites/default/files/ENERGY%20STAR%20Computers%20Final%20Version%208.0%20Specification%20-%20Rev.%20April%202020_0.pdf.

5. TCO Certified : la certification mondiale de durabilité pour les produits informatiques. [En ligne] https://tcocertified.com/french/.

6. 80 PLUS : Program details information. [En ligne] https://www.clearesult.com/80plus/program-details-information#what-is-80plus-specification.

7. IEEE Std 1680-2006. [En ligne] https://standards.ieee.org/standard/1680-2006.html.

8. ISO 26000 Social Responsibility. [En ligne] https://www.iso.org/iso-26000-social-responsibility.html.

9. ISO 14001:2015 Systèmes de management environnemental. [En ligne] https://www.iso.org/fr/standard/60857.html.

10. ISO 50001 Management de l'énergie. [En ligne] https://www.iso.org/fr/iso-50001-energy-management.html.

11. Journal officiel de l'Union européenne : Directive 2015/863. [En ligne] https://eur-lex.europa.eu/legal-content/FR/TXT/?uri=CELEX:32015L0863.

12. https://ec.europa.eu/info/energy-climate-change-environment/standards-tools-and-labels/products-labelling-rules-and-requirements/energy-label-and-ecodesign/product-database/qr-code-new-energy-label_en. [En ligne]

13. A European Green Deal. [En ligne] https://ec.europa.eu/info/strategy/priorities-2019-2024/european-green-deal_en.

14. Loi n° 2010-788 du 12 juillet 2010, Article 75. [En ligne] https://www.legifrance.gouv.fr/jorf/article_jo/JORFARTI000022470999 .

15. Dans la fabrique de « l'indice de réparabilité » en vigueur depuis janvier 2021. [En ligne] https://theconversation.com/dans-la-fabrique-de-lindice-de-reparabilite-en-vigueur-depuis-janvier-2021-155536.

16. Loi climat : et si les ingénieurs en informatique étaient formés à l'écoconception logicielle ? [En ligne] https://www.numerama.com/tech/700900-loi-climat-et-si-les-ingenieurs-en-informatique-etaient-formes-a-lecoconception-logicielle.html.

17. Formosa 1 Offshore Wind Farm. [En ligne] https://www.power-technology.com/projects/formosa-1-offshore-wind-farm/.

18. 'Largest Ever': Google Announces 1.6GW of Renewables Purchases. [En ligne] https://www.greentechmedia.com/articles/read/google-announces-1600-megawatt-in-renewables-purchases.

19. Microchip Giant TSMC Signs 'World's Largest' Corporate Renewables Deal — for Offshore Wind. [En ligne] https://www.greentechmedia.com/articles/read/orsted-signs-worlds-largest-corporate-ppa.

20. Le champ d'éoliennes d'OVH finit de pousser : 6,5 MW à partir de mi-juillet. [En ligne] https://www.nextinpact.com/article/6868/80571-le-champ-deoliennes-dovh-finit-pousser-65-mw-a-partir-mi-juillet.

21. Neutre en carbone depuis 2007.Objectif zéro carbone d'ici 2030. [En ligne] https://sustainability.google/intl/fr/commitments-europe/#.

22. Energy and Carbon Emissions in Neural Architecture Search and Large Model Training. [En ligne] https://scalelite.app.wopla.io/playback/presentation/2.0/playback.html?meetingId=782f0e4459fbb4c3bb136a78cedb620b0c8fc00e-1623156207487.

23. Amazon Durabilité. [En ligne] https://durabilite.aboutamazon.fr/.

24. L'énergie de fabrication est trop souvent négligée. [En ligne] https://www.letemps.ch/opinions/lenergie-fabrication-souvent-negligee.

25. Microsoft reveals the environmental impact of undersea datacenter. [En ligne] https://mspoweruser.com/microsoft-reveals-the-environmental-impact-of-undersea-datacenter/.

26. Pour mieux refroidir ses serveurs, Microsoft les plonge dans un liquide bouillant. [En ligne] https://www.usine-digitale.fr/article/pour-mieux-refroidir-ses-serveurs-microsoft-les-plonge-dans-un-liquide-bouillant.N1079744.

27. Scaleway vaut améliorer l'efficacité énergétique de ses centres de données. [En ligne] https://www.clubic.com/pro/actualite-365850-consommation-d-eau-des-datacenters-scaleway-tire-la-sonnette-d-alarme.html.

28. The cascade effect - Page 10. [En ligne] https://www.uk.insight.com/content/dam/insight/EMEA/uk/shop/emerson/energy-logic.pdf.

29. International Solid-State Circuits Conference 2021 (Intervention Mark Liu / TSMC). [En ligne] https://www.youtube.com/watch?v=8mI8l7jQzHg.

30. Power comsumption SSD SATA. [En ligne] https://qastack.fr/superuser/589709/power-consumption-ssd-vs-hdd.

31. Server virtualization - Page 8. [En ligne] https://www.uk.insight.com/content/dam/insight/EMEA/uk/shop/emerson/energy-logic.pdf.

32. Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes. [En ligne] https://ieeexplore.ieee.org/document/8705815.

33. How does docker affect energy consumption? [En ligne] https://www.sciencedirect.com/science/article/abs/pii/S0164121218301456?via%3Dihub.

34. Acar, Hayri. https://tel.archives-ouvertes.fr/tel-01724069/document. [En ligne] 2018.

35. The Principles of Sustainable Software Engineering. [En ligne] https://docs.microsoft.com/en-us/learn/modules/sustainable-software-engineering-overview/.

36. Mohankumar Muthu, K. Banuroopa and S. Arunadevi. Green and Sustainability in Software Development Lifecycle Process. [En ligne] https://www.intechopen.com/books/sustainability-assessment-at-the-21st-century/green-and-sustainability-in-software-development-lifecycle-process. DOI: 10.5772/intechopen.88030.

37. Energy Efficiency across Programming Languages : How Do Energy, Time, and Memory Relate? [En ligne] 2017. https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf.

38. GitHub - greensoftwarelab / Energy-Languages. [En ligne] https://github.com/greensoftwarelab/Energy-Languages.

39. Texas Instrument - INA260 data sheet. [En ligne] https://www.ti.com/product/INA260.

40. [En ligne] https://github.com/torvalds/linux/blob/master/tools/perf/design.txt.

41. Linux perf Examples. [En ligne] http://www.brendangregg.com/perf.html.

42. Linux Systems Performance Tracing, Profiling, and Visualization. [En ligne] https://indico.cern.ch/event/980497/contributions/4130271/.

43. Julia Evans - Profiling & Tracing with perf. [En ligne] https://jvns.ca/blog/2018/04/16/new-perf-zine/.

VII. Annexes

A. Synthèse des écolabels

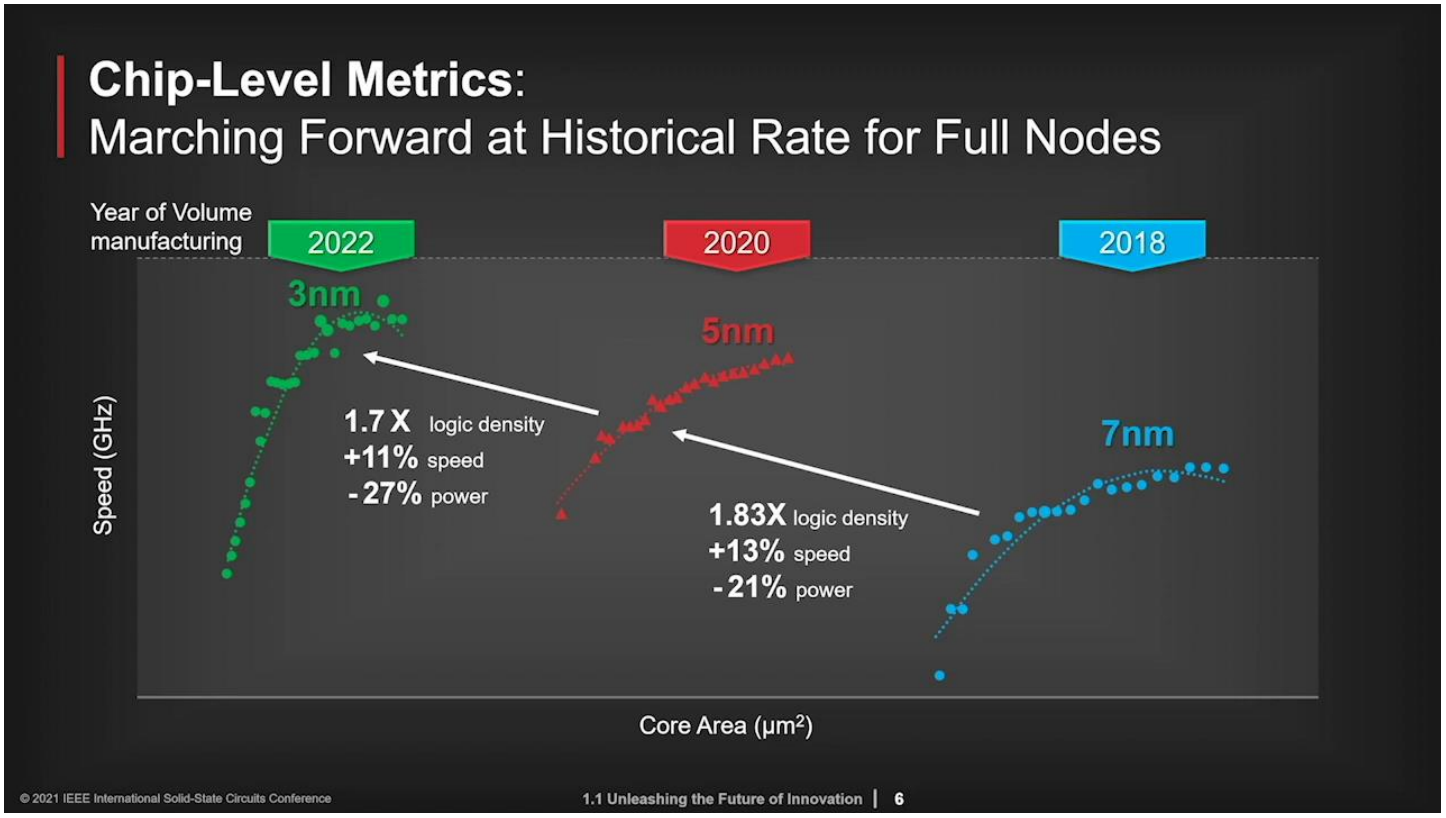
Nom	Création	GEN*	Région	Objectif	Obtention
80 Plus	2004		USA	Efficacité énergétique	Vérifié par une organisation indépendante respectant les standards IEEE
ABNT Ecolabel	1993	ü	Brésil	Cycle de vie	Vérifié par une organisation indépendante
Blue Angel	1977	ü	Allemagne	Réduire impact sur environnement sans perte de qualité, fiabilité et sécurité	Jury de 13 membres
China Energy Label	1999		Chine	Évaluation impact au long du cycle de vie pour obtenir une note	Tests réalisés par des laboratoires
China Environmental Labelling	1993	ü	Chine	Standards qualité et protection de l'environnement durant tout le cycle de vie	Vérifié par une organisation indépendante
Climatop	2008		Suisse	Récompense le produit le plus respectueux pour l'environnement pour chaque type de produit	Vérifié par une organisation indépendante
Eco-leaf	2002		Japon	Évaluation impact au long du cycle de vie	Vérifié par une organisation indépendante
EcoLogo	2010	ü	Amérique du Nord	Évaluation impact au long du cycle de vie	Vérifié par une organisation indépendante ayant des accréditation ISO
Ecomark:india	1991		Inde	Évaluation impact au long du cycle de vie	Vérifié par une organisation indépendante
Ecomark:Japan	1989	ü	Japon	Attribué aux produits meilleurs que leurs concurrents sur leur cycle de vie	Vérifié par une organisation indépendante
Ekolabel: Indonesia	2009	ü	Indonésie	Évaluation impact au long du cycle de vie	Vérifié par une organisation indépendante
Emblem of Guarantee of Environmental Quality: Catalonia	1994		Catalogne	Évaluation impact au long du cycle de vie	Vérifié par une organisation indépendante
Energy Rating Programme: Australia	1992		Australie	Évaluation impact au long du cycle de vie pour obtenir une note	Vérifié par une organisation indépendante
Energy Saving Labeling Program: Japan	2000		Japon	Diminution de la consommation électrique lors de l'utilisation	Vérifié par une organisation indépendante
Energy Saving Recommended	1992		Angleterre	Efficacité énergétique	Vérifié par la société produisant le produit
Environmentally Friendly Label: Croatia	1993		Croatie	Attribué aux produits meilleurs que leurs concurrents sur leur cycle de vie	Vérifié par une organisation indépendante ayant des accréditations ISO
EPEAT	2006		43 pays	Évaluation impact au long du cycle de vie	Vérifié par une organisation indépendante ayant des accréditations ISO
EU Ecolabel	1992	ü	Europe	Évaluation impact au long du cycle de vie	Vérifié par une organisation indépendante ayant des accréditations ISO
EU Energie label	1992		Europe	Efficacité énergétique	Vérifié par la société produisant le bien
ECMA: TR/70	1997		Europe	Efficacité énergétique	Vérifié par la société produisant le bien
Future Friendly - Proctor and Gamble	2007		N/A	Privé - Seulement pour produits P&G- Évaluation impact au long du cycle de vie	Vérifié par la société produisant le bien
Green Crane: Ukraine	2002		Ukraine	Attribué aux produits meilleurs que leurs concurrents sur leur cycle de vie	Vérifié par une organisation indépendante

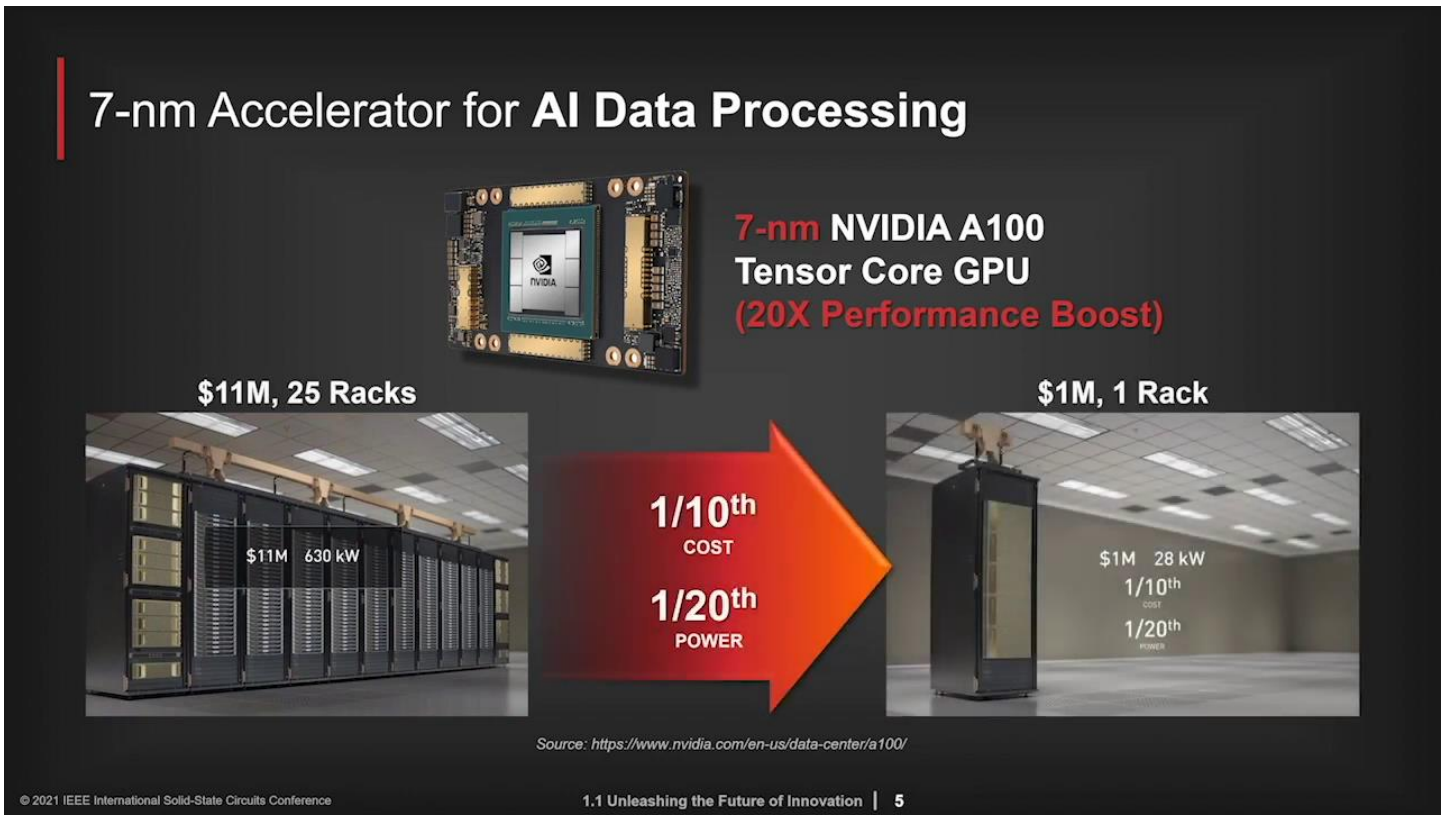
Nom	Création	GEN*	Région	Objectif	Obtention
Green IT	2008		N/A	Privé - Seulement pour produits Fujitsu Siemens. Evaluation impact au long du cycle de vie	Vérifié par la société produisant le bien
Hong Kong Eco-label	1995	ü	Hong-Kong	Efficacité énergétique	Vérifié par une organisation indépendante ayant des accréditation ISO
Korean Ecolabel	1992	ü	Corée du Sud	Efficacité énergétique	Vérifié par une organisation indépendante ayant des accréditation ISO
Label Energy Star	1992		USA, Canada, Japon, Suisse, Taïwan	Efficacité énergétique	Tests en laboratoires indépendants
Phillips Green Logo	1994		N/A	Privé - Efficacité énergétique et recyclage	Vérifié par la société produisant le bien
Singapore Green Label Scheme (SGLS)	1992	ü	Singapour	Évaluation impact au long du cycle de vie. Basé sur d'autres éco-standards	Vérifié par la société produisant le bien et ayant des accréditations ISO
TCO	1992	ü	Suède	Évaluation impact au long du cycle de vie	Tests réalisés par partenaires indépendants
Thai Green Label	1994	ü	Thaïlande	Attribué aux produits meilleurs que leurs concurrents sur leur cycle de vie	Vérifié par une organisation indépendante ayant des accréditations ISO

*GEN : Membres du Global Ecolabelling Network

B. Impact de la finesse de gravures

Ci-dessous, des données issues de la présentation de Mark Liu (TSMC) lors de l'International Solid-State Circuits Conference qui s'est tenue en février 2021 [29].





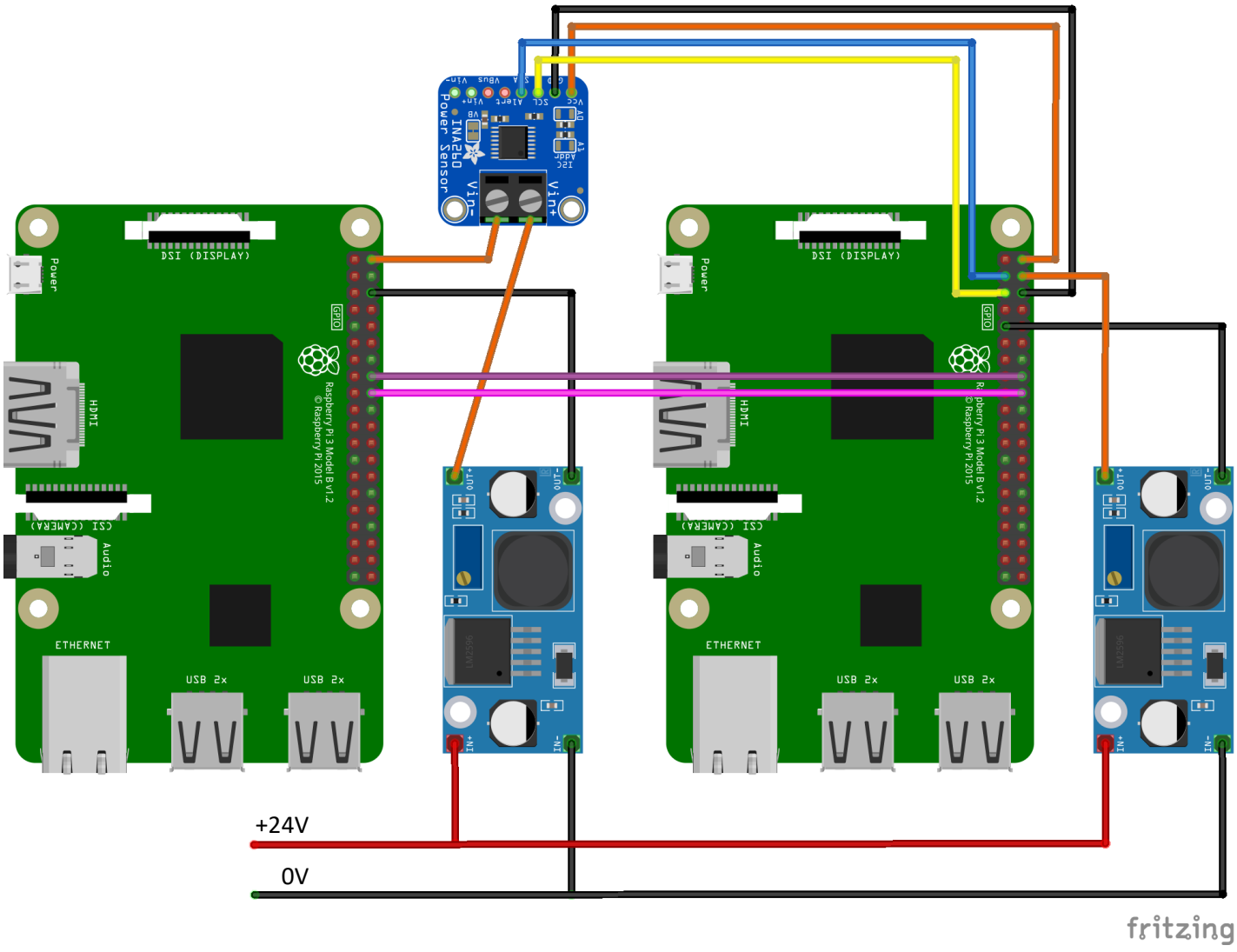
C. Caractéristiques Raspberry PI 3B+

Processeur	quad-core, Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Cache	16Ko L1P (Instruction) 16Ko L1D (Data) 512Ko L2
Mémoire	1Go LPDDR2 SDRAM

Source : Raspberry Pi 3 Model B+ product brief

D. Schéma de câblage de l'expérimentation

Notre système utilise une alimentation 24V « de bureau » d'une capacité de 2A. Des convertisseurs de tensions DC-DC sont donc utilisés pour alimenter les Raspberry par le GPIO en 5V. Ils sont équipés d'un régulateur TI-LM2596 réglable.

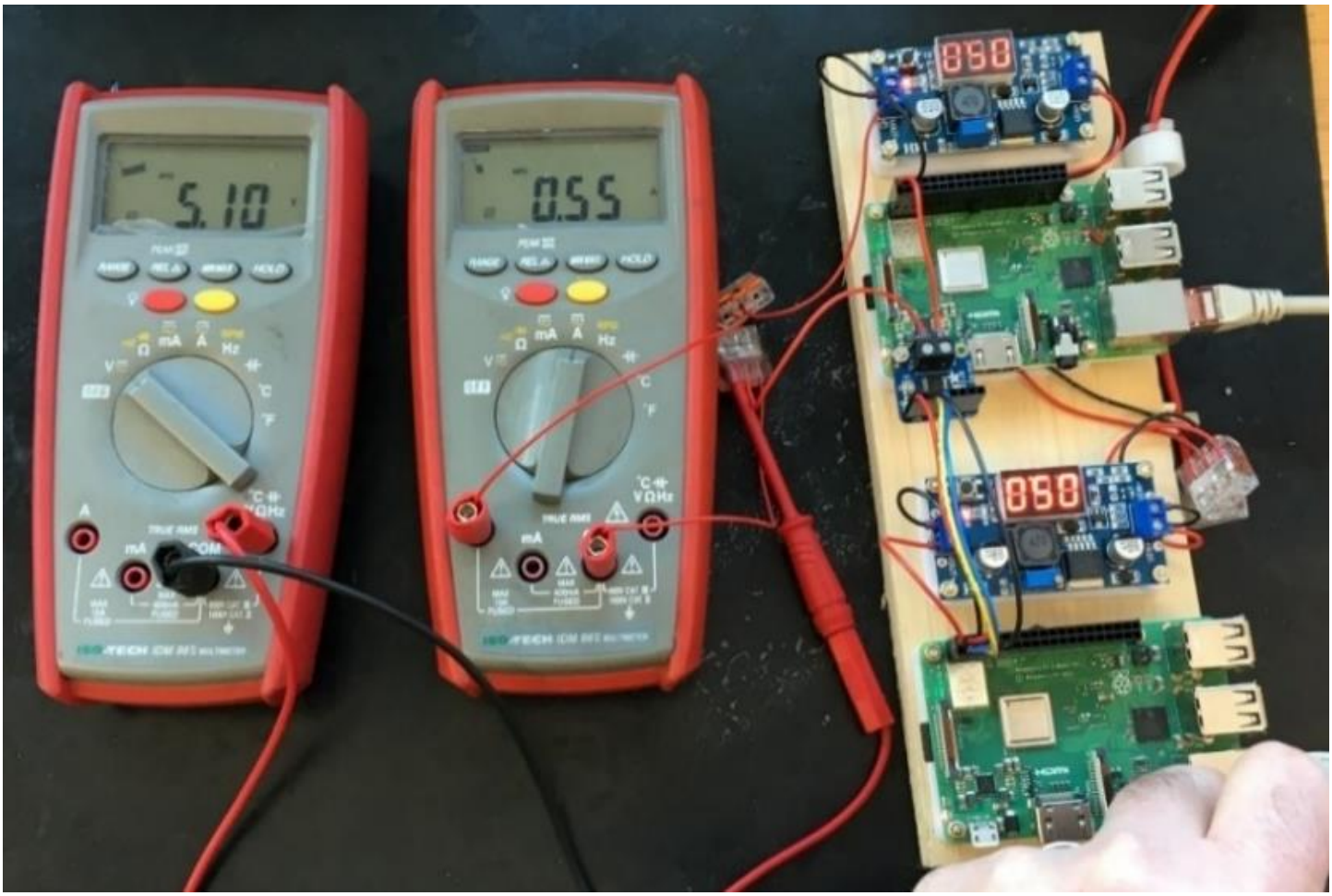


Pin 2 : +5V
Pin 3 [SDA] : I²C/SMBus
Pin 5 [SCL] : I²C/SMBus
Pin 6 : 0V
Pin 9 : 0V
Pin 16 [GPIO23] : *synchro*, permet de démarrer l'enregistrement via le capteur INA260
Pin 18 [GPIO24] : *isRunning*, permet de déterminer si un programme est en cours d'exécution

E. Vérification de la validité des mesures

La vérification de la cohérence des données acquises à l'aide du capteur TI-INA260 a été réalisée à l'aide d'un multimètre et d'un ampèremètre, à l'état de repos et à une charge CPU de 25%, correspondant à l'occupation d'un cœur à 100%.

Ci-dessous la photo du montage de l'expérimentation à une charge CPU de 25% :



F. Algorithmes

1. algo-01

```
Initialiser un tableau vide Résultat

Ouvrir fichier « input.txt »
    Lire la première ligne et récupérer la taille de l'image baseX, baseY
    Pour chaque ligne :
        Charger la ligne lue dans un tableau : baseGrid[indexLigne] ← ligne

Pour y allant de 0 à baseY :
    Pour x allant de 0 à baseX :
        Si baseGrid[y][x] == "#" :
            Ajouter à Résultat « FILL x, y, 1 »

Ouvrir fichier « solution.txt » :
    Écrire chaque ligne de Résultat
```

2. algo-02

```
Initialiser un tableau vide Résultat

Ouvrir fichier « input.txt »
    Lire la première ligne et récupérer la taille de l'image baseX, baseY
    Pour chaque ligne :
        Charger la ligne lue dans un tableau : baseGrid[indexLigne] ← ligne

Pré-Calculer drawableArea pour chaque baseGrid[y][x] == "#"

Pour y allant de 0 à baseY :
    Pour x allant de 0 à baseX :
        Si baseGrid[y][x] == "#" :
            Trouver la drawableArea la plus efficiente :
                Ajouter à Résultat « FILL x, y, d »

Ouvrir fichier « solution.txt » :
    Écrire chaque ligne de Résultat
```

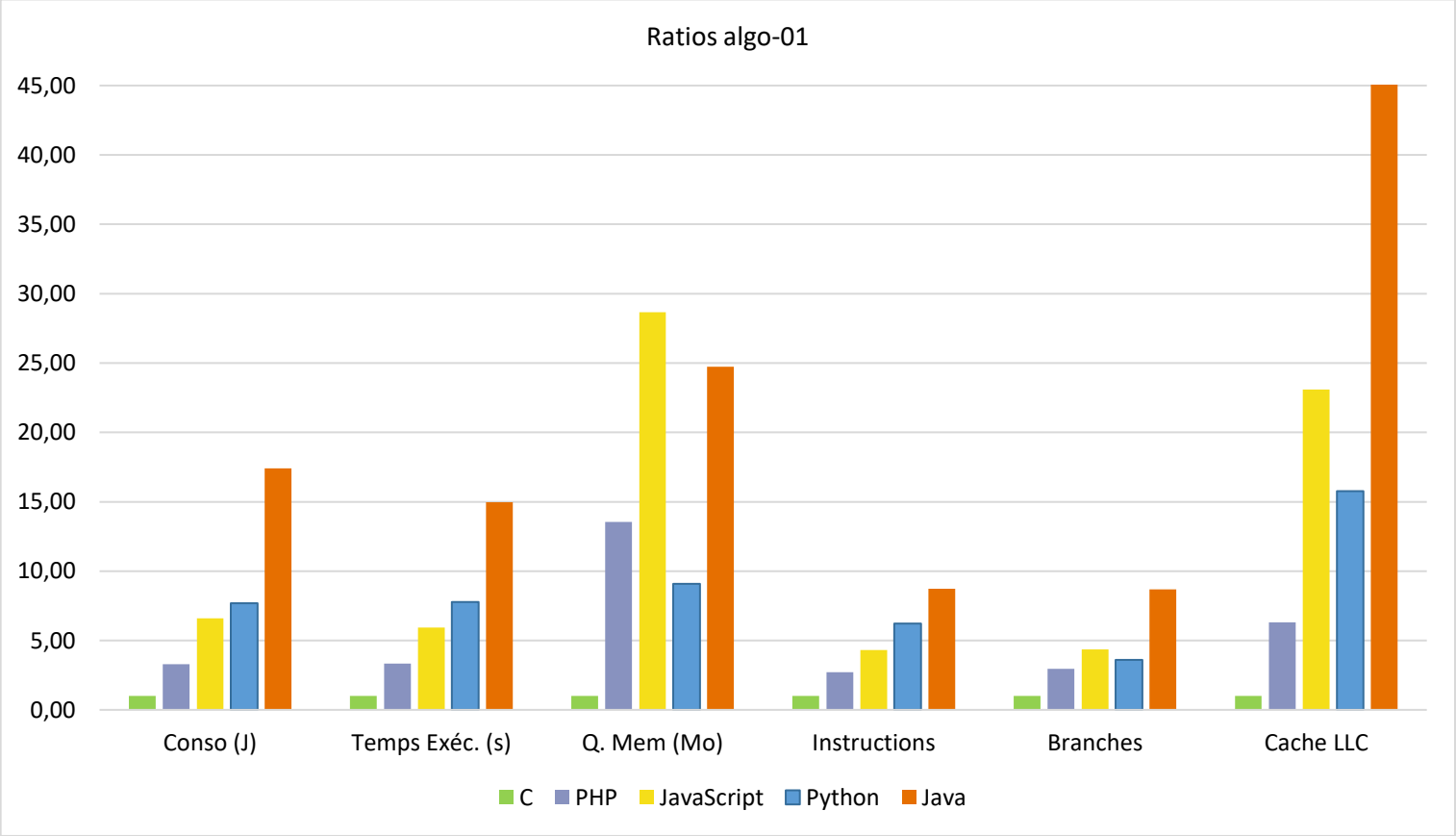
3. algo-03

```
Initialiser un tableau vide Résultat

Ouvrir fichier « input.txt »
    Lire la première ligne et récupérer la taille de l'image baseX, baseY
    Pour chaque ligne :
        Charger la ligne lue dans un tableau : baseGrid[indexLigne] ← ligne

Pré-Calculer drawableArea, isolationIndex pour chaque baseGrid[y][x] == "#"

Pour index allant de max(isolationIndex) à 2 :
    Pour y allant de 0 à baseY :
        Pour x allant de 0 à baseX :
            Si baseGrid[y][x] == "#" et index == isolationIndex[y][x] :
                Trouver drawableArea la plus efficiente :
                    Ajouter à Résultat « FILL x, y, d »

Pour y allant de baseY à 0 :
    Pour x allant de 0 à baseX :
            Si baseGrid[y][x] == "#" :
                Trouver drawableArea la plus efficiente :
                    Ajouter à Résultat « FILL x, y, d »

Ouvrir fichier « solution.txt » :
    Écrire chaque ligne de Résultat
```

G. Synthèses exécutions

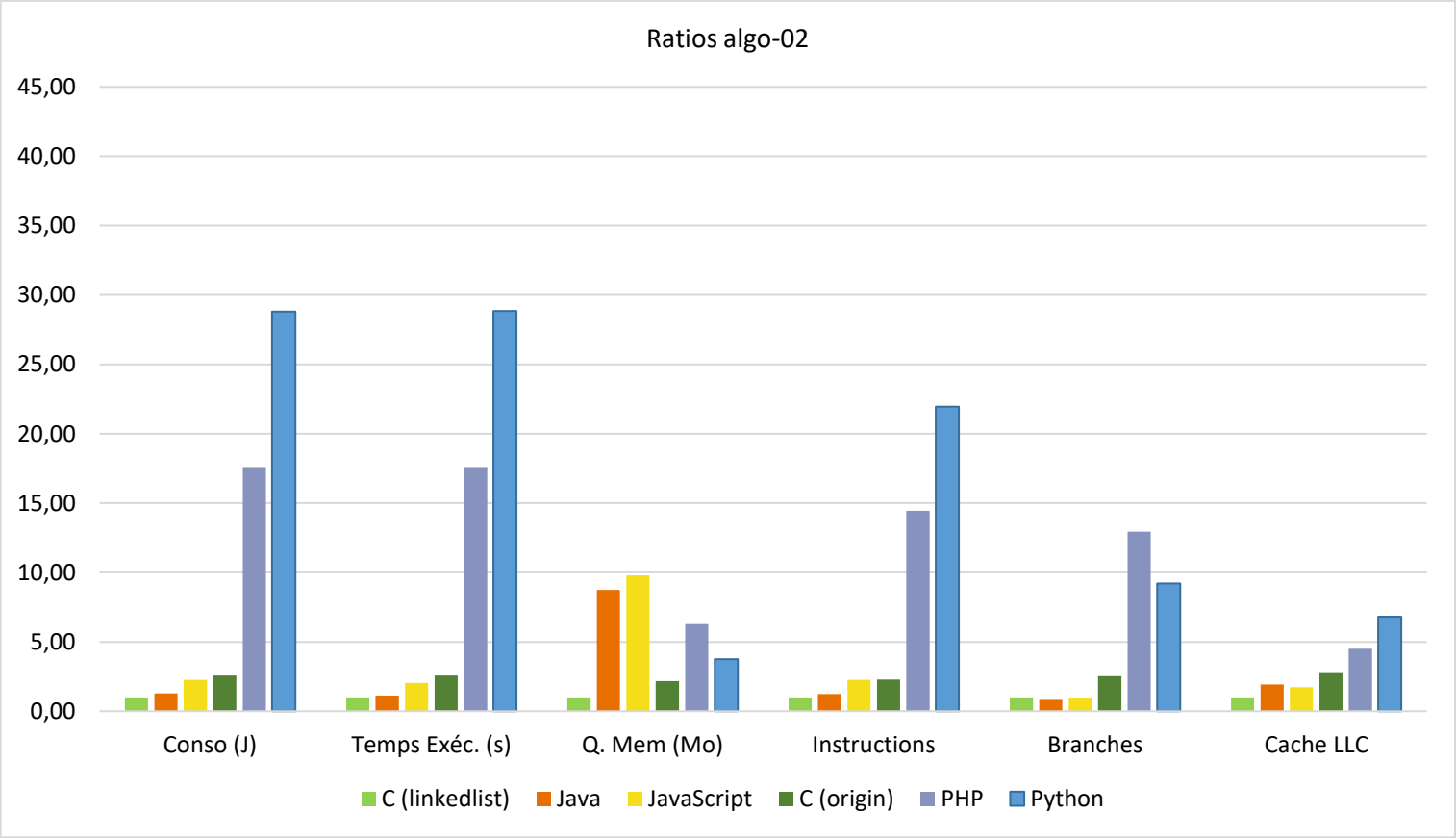
1. Synthèse exécution algo-01

Lang.	*Conso*		*Temps Exéc.*		*Q. Mémoire*		*Instructions*		*Branches*		*Accès Cache LLC*	
	(J)	*Ratio*	*(s)*	*Ratio*	*(Mo)*	*Ratio*	*Ratio*	*IPC brut*	*Ratio*	*%déf. brut*	*Ratio*	*%déf. brut*
C	0.10	1.00	0.11	1.00	1.39	1.00	1.00	0.87	1.00	5.82	1.00	3.74
PHP	0.32	3.30	0.37	3.33	18.90	13.55	2.72	0.71	2.96	5.46	6.31	10.85
JavaScript	0.63	6.60	0.66	5.93	39.95	28.66	4.30	0.63	4.35	9.57	23.10	12.67
Python	0.73	7.69	0.86	7.77	12.65	9.08	6.22	0.69	3.60	17.73	15.77	7.66
Java	1.66	17.41	1.66	14.96	34.48	24.74	8.74	0.51	8.67	27.61	76.57	7.63



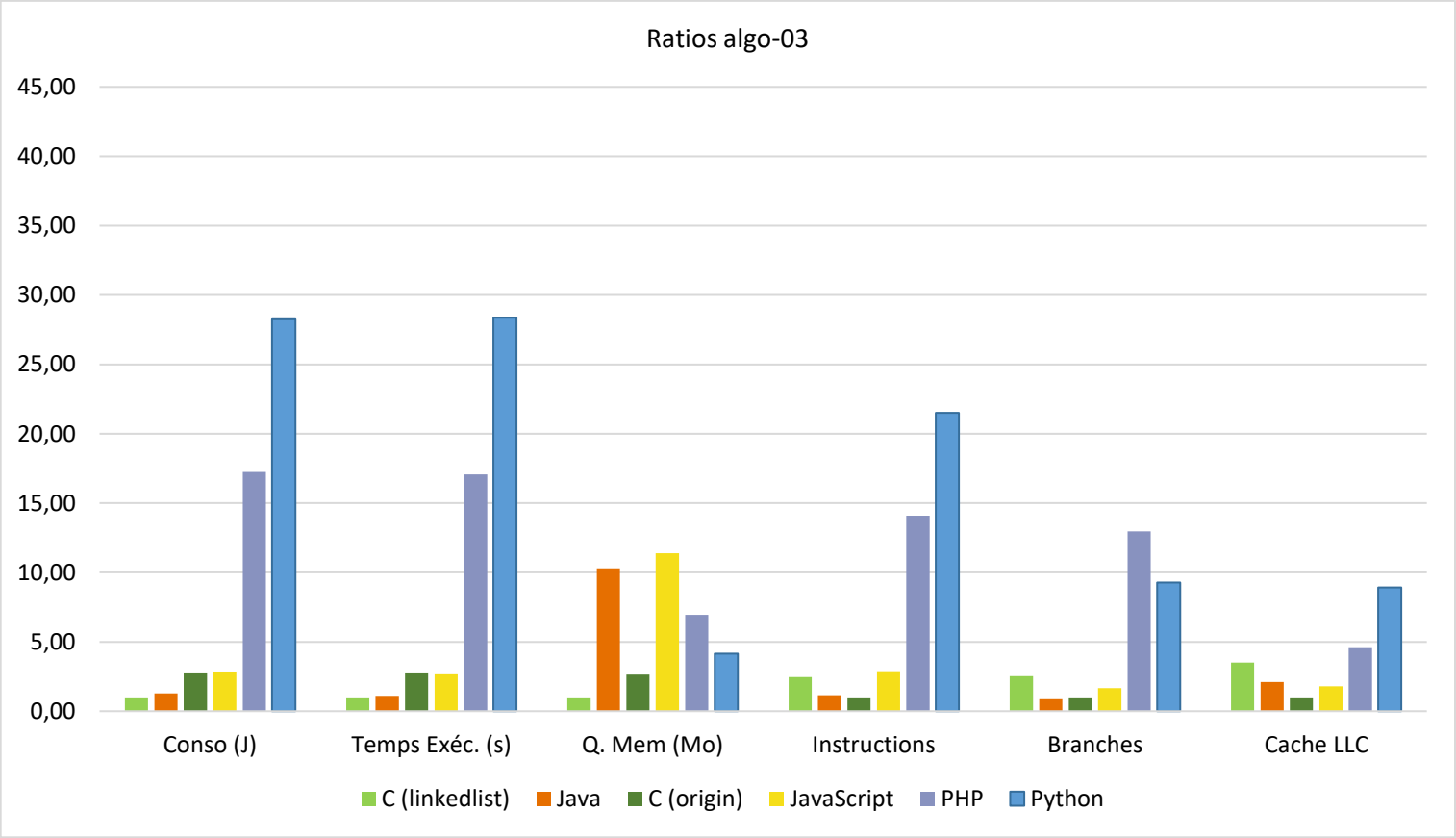
2. Synthèse exécution algo-02

Lang.	Conso		Temps Exéc.		Q. Mémoire		Instructions		Branches		Accès Cache LLC	
	(J)	Ratio	(s)	Ratio	(Mo)	Ratio	Ratio	IPC brut	Ratio	%déf. brut	Ratio	%déf. brut
C (linkedlist)	14.13	1.00	16.15	1.00	8.55	1.00	1.00	0.90	1.00	0.30	1.00	49.77
Java	18.09	1.28	18.10	1.12	3.94	8.75	1.23	0.99	0.81	5.93	1.93	41.17
JavaScript	32.12	2.27	32.88	2.04	34.46	9.78	2.27	1.00	0.96	3.32	1.73	46.79
C (origin)	38.49	2.58	41.68	2.58	38.51	2.17	2.28	0.79	2.53	0.76	2.82	49.96
PHP	248.96	17.61	284.28	17.60	24.78	6.29	14.44	0.74	12.93	2.31	4.51	43.18
Python	407.36	28.81	466.08	28.86	14.72	3.74	21.94	0.68	9.20	33.50	6.81	33.91



3. Synthèse exécution algo-03

Lang.	Conso		Temps Exéc.		Q. Mémoire		Instructions		Branches		Accès Cache LLC	
	(J)	Ratio	(s)	Ratio	(Mo)	Ratio	Ratio	IPC brut	Ratio	%déf. brut	Ratio	%déf. brut
C (linkedlist)	14.46	1.00	16.60	1.00	3.96	1.00	1.00	0.89	1.00	0.73	1.00	49.92
Java	18.71	1.29	18.35	1.11	40.76	10.30	1.15	0.93	0.87	6.69	2.10	38.28
C (origin)	40.45	2.80	46.44	2.80	10.40	2.63	2.46	0.79	2.52	0.32	3.51	49.67
JavaScript	41.44	2.87	44.09	2.66	45.04	11.39	2.88	0.97	1.66	2.28	1.79	45.33
PHP	249.35	17.24	283.35	17.07	27.51	6.95	14.08	0.74	12.97	2.38	4.62	42.04
Python	408.64	28.26	470.85	28.37	16.37	4.14	21.51	0.68	9.28	33.39	8.92	27.80



4. Synthèse exécution algo-03-r1

Lang.	*Conso*		*Temps Exéc.*		*Q. Mémoire*		*Instructions*		*Branches*		*Accès Cache LLC*	
	(J)	*Ratio*	*(s)*	*Ratio*	*(Mo)*	*Ratio*	*Ratio*	*IPC brut*	*Ratio*	*%déf. brut*	*Ratio*	*%déf. brut*
C	2.74	1.00	4.04	1.00	5.76	1.00	1.00	0.83	1.00	1.05	1.00	12.24
JavaScript	4.51	1.65	5.62	1.39	44.22	7.68	1.38	0.83	1.19	7.26	2.91	12.26
Java	16.37	5.97	22.14	5.48	40.87	7.10	6.08	0.92	5.15	7.56	8.92	7.34
PHP	46.77	17.06	55.49	13.74	27.45	4.77	12.09	0.73	13.24	5.48	10.56	11.53
Python	104.85	38.25	125.18	30.99	24.51	4.25	24.52	0.66	12.53	26.95	41.52	6.05