</> ITN

MINES PARIS | PSL

30 ans après « Un plaidoyer pour l'allègement du logiciel »

A plea for Lean Software (N. Wirth, 1995)

Georges-André Silber
https://mines.paris/p/gas

14 avril 2026

Chaire « logiciel responsable », projets PACE et CAMELIA
Centre de recherche en informatique (CRI)
Mines Paris — PSL

Faire mieux avec moins

Logiciel responsable ?

Responsabilité environnementale

- consommer moins d'énergie lors de l'utilisation des logiciels
- utiliser plus efficacement le matériel, plus longtemps
- générer moins de déchets

Responsabilité sociétale

- contribuer aux logiciels libres, biens communs numériques
- produire des logiciels plus sûrs

Contexte environnemental : un dépassement des limites planétaires

Changement climatique : deux limites dépassées

07 Les Echos Vendredi 25 et samedi 26 octobre 2024

Sans effort supplémentaire, le réchauffement atteindra 3,1 °C

ENVIRONNEMENT

Le programme des Nations unies pour l'environnement a publié jeudi son rapport sur les émissions de gaz à effet de serre.

Les engagements des Etats membres ne permettent pas de respecter l'objectif de l'Accord de Paris.

ce rapport très attendu chaque année montre que la poursuite des politiques actuelles de réduction des émissions de gaz à effet de serre conduirait à un réchauffement de 3,1 °C d'ici à la fin du siècle, par rapport à l'ère préindustrielle.

Stratégies nationales insuffisantes

Un écart béant par rapport à l'Accord de Paris. En 2015, les pays membres de l'ONU s'étaient engagés à limiter le réchauffement sous la barre des 2 °C, et à tout faire pour se rapprocher de 1,5 °C.

« Chaque fraction de degré évitée est cruciale, en termes de vies sau-

« Leurs engagements inconditionnels entraîneraient une baisse des émissions de 4 % d'ici à 2030, ce qui placerait la planète sur la trajectoire d'un réchauffement de 2,8 °C en 2100. Et ce recul passerait à 18 % en ajoutant leurs engagements sous conditions, de recevoir un soutien financier par exemple, ce qui correspondrait à un réchauffement de 2,6 °C », explique Joeri Rogelj, l'un des auteurs du rapport. Or pour espérer un réchauffement limité à 2 °C, la baisse devrait atteindre 28 % (et 42 % pour 1,5 °C).

Les émissions

Les auteurs ont pourtant identifié des solutions permettant de rester sur la trajectoire de l'objectif de l'Accord de Paris, à un coût inférieur à 200 dollars la tonne de CO_2 évitée. *« Le potentiel de réduction est de 31 gigatonnes à 2030 et 41 gigatonnes à 2035 »*, écrivent-ils.

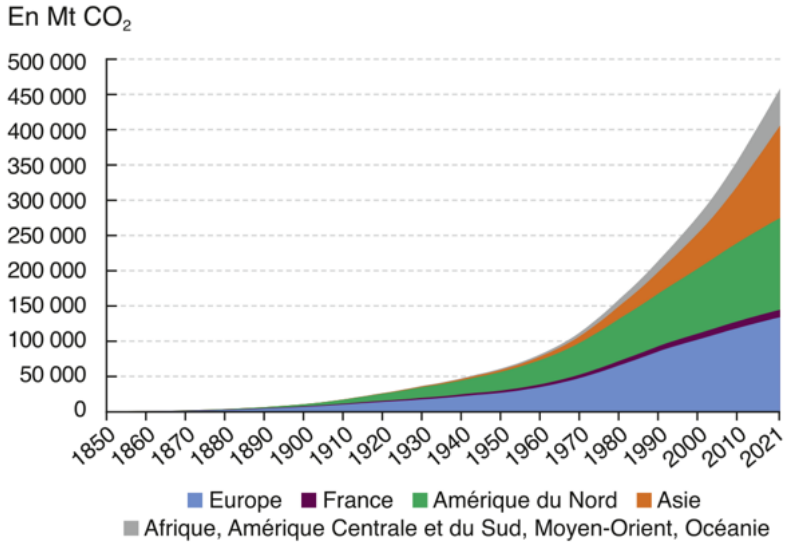
Multiplier par six les investissements

Et ce notamment grâce au développement d'électricité solaire photovoltaïque et éolienne, à des actions sur la forêt (meilleure gestion, arrêt de la déforestation, etc.), ou à d'autres mesures comme les économies d'énergie et une électrification

Concentration de CO_2 : lim. **350** ppm → **379** ppm (2005), **425** ppm (2023)

↑ du forçage radiatif : lim. **+1, 0** W/m^2 → **+1, 66** W/m^2 (2005), **+2, 72** W/m^2 (2023)

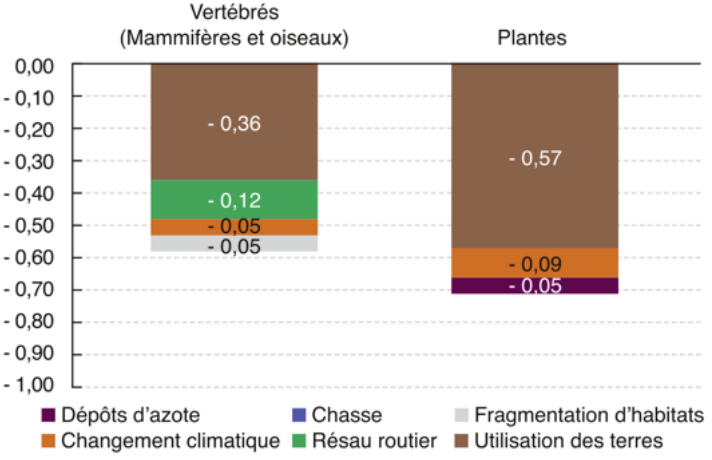
Cumuls des émissions de CO_2 de 1850 à 2021



Source : Global Carbon Project, via https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/la-france-face-aux-neuf-limites-planetaires/4-changement-climatique.

Érosion de la biodiversité : IIB et extinction, limites dépassées

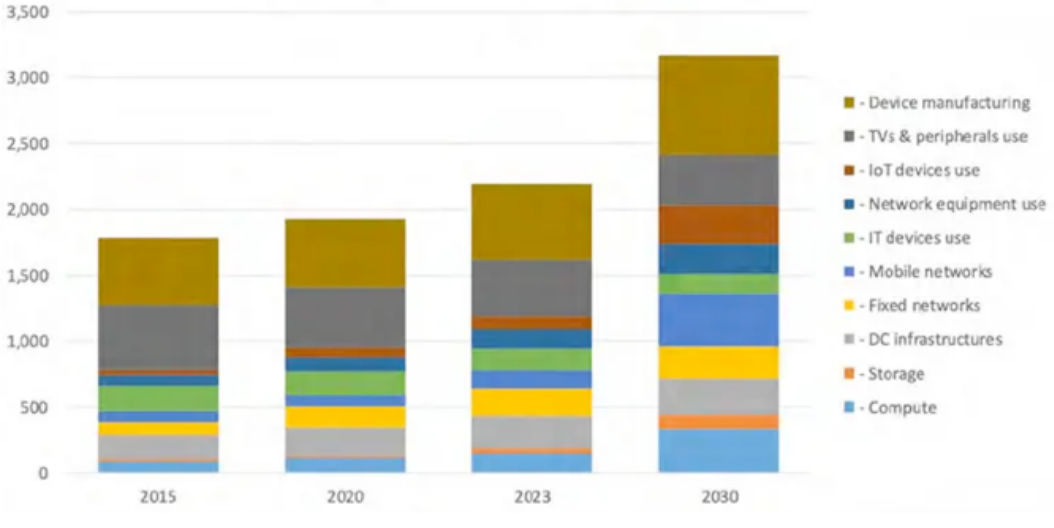
Pressions contribuant à la perte de l'abondance moyenne des espèces (MSA), en France métropolitaine, en 2020 (abondance moyenne perdue, 1-MSA)



Source : modèle GLOBIO, via https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/la-france-face-aux-neuf-limites-planetaires/5-erosion-de-la-biodiversite.

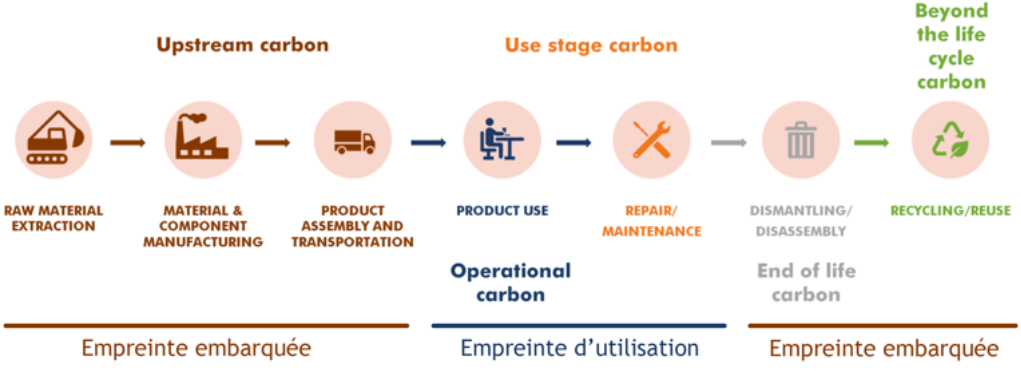
L’impact environnemental croissant du numérique

Utilisation d'énergie dans l'industrie du numérique



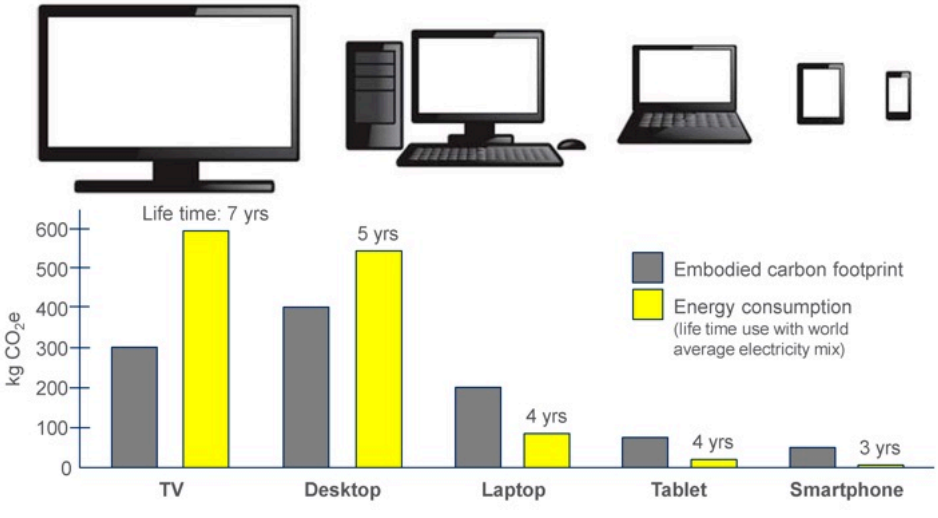
Source : L. Eeckhout, «*Sustainable computer systems*», HiPEAC 2024 Keynote 2, 23 février 2024.
https://www.youtube.com/watch?v=jUw7QQ-a00Q

Empreintes environnementales *embarquée* et *d'utilisation* (1)



Source : L. Belkhir et al., «*Assessing ICT global emissions footprint : Trends to 2040 & recommendations*», J. of Cleaner Production, 3/2018.

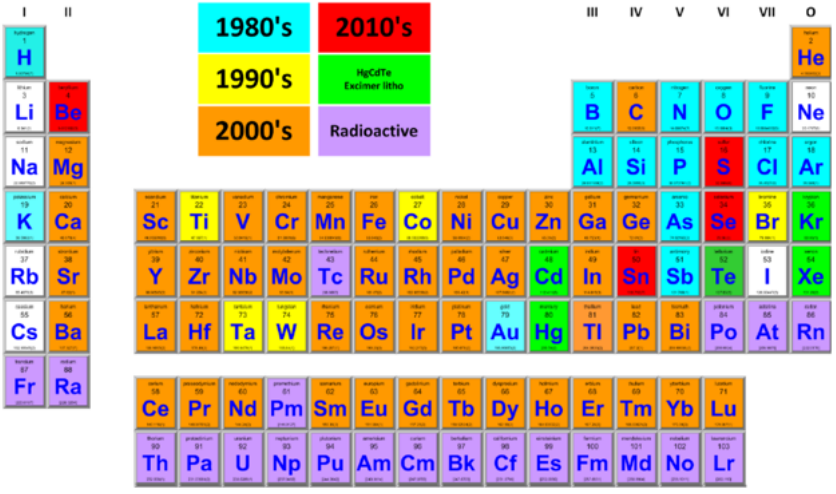
Empreintes environnementales *embarquée* et *d'utilisation* (2)



Source : J. Malmodin et D. Lundén, «*The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010–2015*», Sustainability, 9/2018.

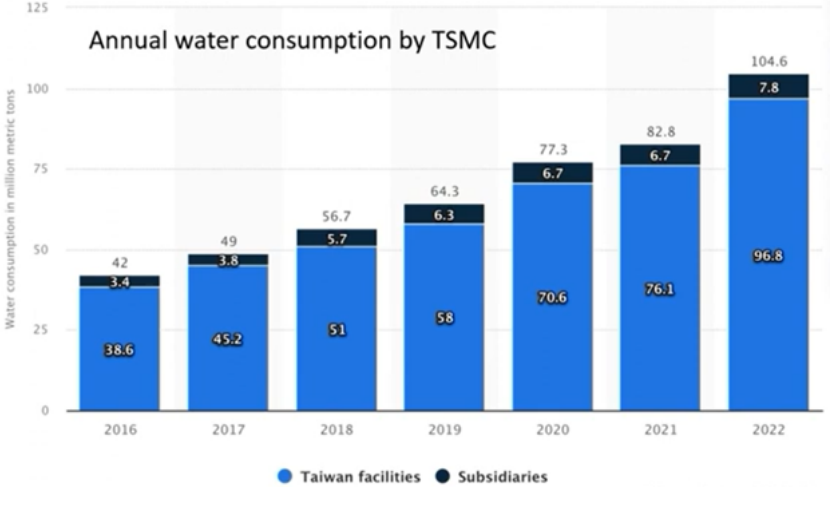
Crédit : Arnout Fierens (`https://arnulf.be`).

Utilisation des éléments chimiques dans les circuits intégrés



Source : T. Ernst, L.-Å. Ragnarsson, et J.-P. Raskin, «*Sustainable materials and production*», HiPEAC Vision 2024, janv. 2024.

Utilisation d'eau pure par l'industrie des semi-conducteurs



Source : L. Eeckhout, «*Sustainable computer systems*», HiPEAC 2024 Keynote 2, 23 février 2024.
https://www.youtube.com/watch?v=jUw7QQ-a00Q

En même temps : nous remplaçons des machines fonctionnelles

- Laboratoire expérimental
- 100 machines
- 169 CPU, 1 094 cœurs
- 2,234 To RAM / 65 To stockage
- En état de marche
- Considérées dépassées
- Q1 : Prolonger leur vie (fiabilité) ?
- Q2 : Énergie ?
- Q3 : Performance ?
- → Utilisation pour nos mesures

Contexte technologique : *bloatware*

Niklaus Wirth, *A Plea for Lean Software*, 1995

Software expands to fill the available memory.
— Parkinson

Software is getting slower more rapidly than hardware is becoming faster.
— Reiser

Source : https://dl.acm.org/doi/10.1109/2.348001. Voir également : https://www.enseignementsup-recherche.gouv.fr/ressources-pedagogiques/notice/view/oai%253Awww.unit.eu%253Aunit-ori-wf-1-6785

Nathan Myhrvold (Microsoft), 1997

Le logiciel est comme un gaz, il se répand autant que le permet son support. Corollaire, le logiciel se développe jusqu'à ce qu'il soit limité par la loi de Moore. Il n'atteindra jamais un stade de maturité industrielle. L'industrie du logiciel est et restera toujours en état de crise.

En fait, c'est la raison pour laquelle il existe un marché pour des processeurs plus rapides — les concepteurs de logiciels ont toujours consommé de nouvelles capacités aussi vite ou plus vite que les concepteurs de puces ne pouvaient les mettre à disposition.

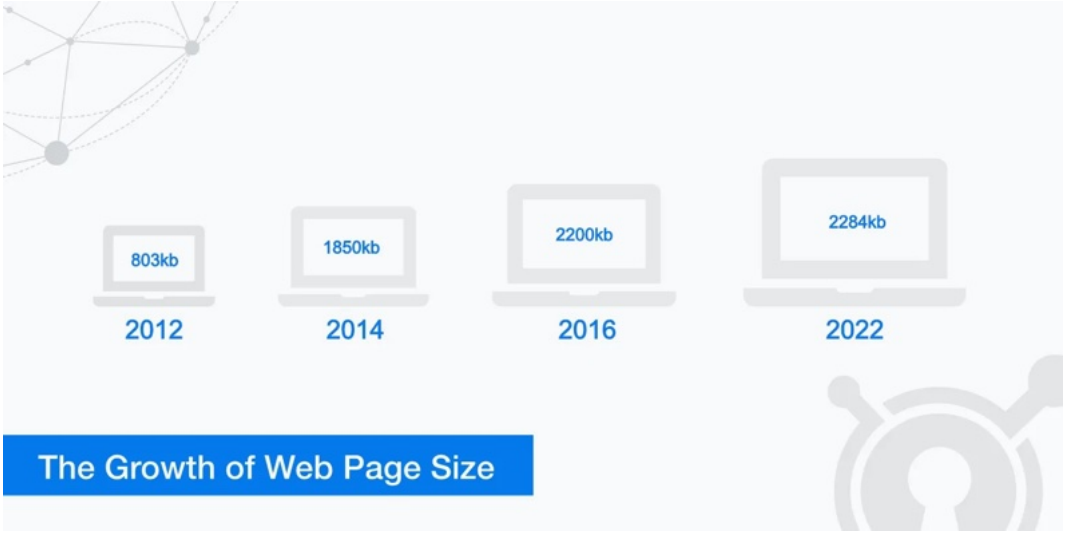
Source : "Nathan Myhrvold - The future of software, the software industry, and Windows '47", ACM 97, the next 50 years of computing.
https://www.youtube.com/watch?v=gE4N0G9kElM

Édition du fichier de cette présentation

`presentation.tex` : 22 126 octets ≡ 22 Ko

Logiciel	Occupation mémoire (ko)
vim	3 500 Ko
nano	3 800 Ko
neovim	10 300 Ko
emacs	48 000 Ko
textedit	81 200 Ko
codium	475 000 Ko
zed	1 600 000 Ko
pycharm	3 440 000 Ko

Évolution de la taille moyenne des pages web



Contexte : la qualité du logiciel

En novembre 1988, un virus informatique a attaqué les ordinateurs connectés à l'Internet encore naissant. Le virus exploitait une erreur de programmation : il supposait que l'on pouvait faire confiance à un autre ordinateur pour envoyer la bonne quantité de données. Il s'agissait d'une simple erreur et la correction était triviale, mais le langage de programmation utilisé était vulnérable à ce type d'erreur et il n'existait pas de méthodologie standard pour détecter ce genre de problème.

— Adam Barr, *The Problem with Software* (2018)

En avril 2014, un virus informatique a attaqué les ordinateurs connectés à l'Internet encore naissant. Le virus exploitait une erreur de programmation : il supposait que l'on pouvait faire confiance à un autre ordinateur pour envoyer la bonne quantité de données. Il s'agissait d'une simple erreur et la correction était triviale, mais le langage de programmation utilisé était vulnérable à ce type d'erreur et il n'existait pas de méthodologie standard pour détecter ce genre de problème.

— Adam Barr, *The Problem with Software* (2018)

Programmation logicielle : pas (encore) de l'ingénierie

- Marqueurs de l'ingénierie : solidité, durabilité, fiabilité, performance, efficacité, réparabilité, évolutivité…
- Après plus de 60 ans de développement logiciels :
- « *vulnerable programming language* »
- « *no way to detect mistakes* »
- Résultat : quantité de bugs visibles par les utilisateurs, la réinvention de la roue en permanence, des délais et budgets peu prévisibles.
- D'autres disciplines ont su apprendre de leurs erreurs (aviation)
- … et produire un corpus de connaissances et de méthodes.

Éducation des programmeurs

- L'éducation d'un programmeur : écriture de petits programmes
- Peu de préparation à la création de logiciels à grande échelle
- Mythe du héros, épreuves de codage pour l'embauche, hackathons
- Peu de réponse à : « *qu'est-ce qu'un bon logiciel ?* »
- Un diplôme en informatique ne garantit pas un savoir précis
- Très différent d'autres disciplines (médecine)
- Adam Barr : « *Is software development really hard, or are software developers not that good at it ?* »

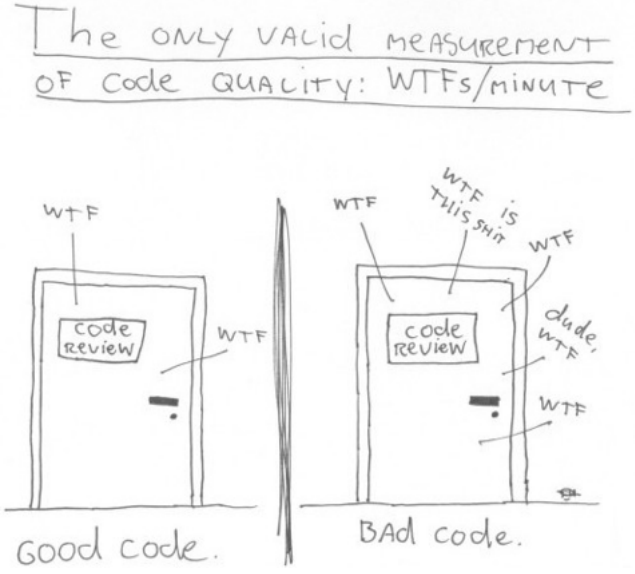
Hardware vs Software

L'ingénierie logicielle est victime de cette fausse idée :

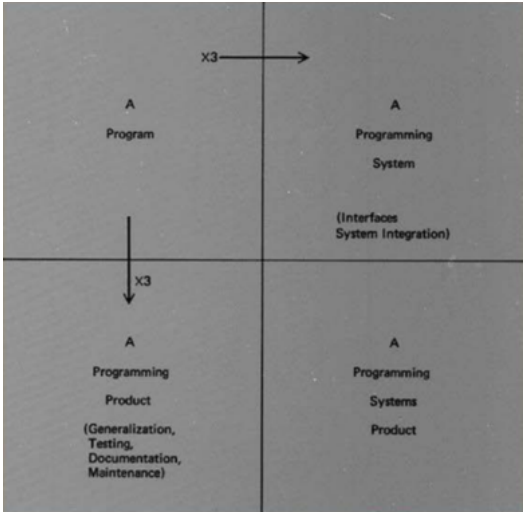
« *Hardware is so-termed because it is **hard** or rigid with respect to changes, whereas software is **soft** because it is easy to change.* »

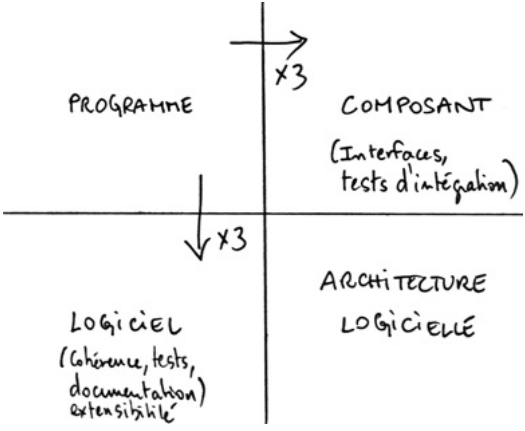
— Wikipedia, article « *Computer hardware* »

WTFs / minute



F.P. Brooks : "The Mythical Man-Month" (1975)





Sécurité des logiciels libres vs propriétaires ?

La disponibilité du code source ne garantit pas la sécurité,

mais les logiciels libres ont bien un avantage structurel décisif en matière de sécurité informatique :

la capacité de modifier celui-ci garantit au moins la possibilité d'obtenir un correctif en cas d'exposition à une vulnérabilité.

Par contre, les **logiciels libres souffrent d'un manque de financement** : "*... the* (XZ Utils) *incident also started a discussion regarding the viability of having critical pieces of cyberinfrastructure depend on unpaid volunteers.*"

(voir `https://en.wikipedia.org/wiki/XZ_Utils_backdoor`).

Le logiciel : un fort potentiel d'amélioration

Leiserson (2020) : *"there is plenty of room at the top"*

MINES PARIS | PSL

```
# Exemple: multiplication de matrices
for i in range(4096):
  for j in range(4096):
    for k in range(4096):
      C[i][j] += A[i][k] * B[k][j]
```

#	Method	Time (s)	Gflop/s	Speedup	Rel. sp.	% peak
1	Python	25 552,48	0,005	1	—	0,00
2	Java	2 372,68	0,058	11	10,8	0,01
3	C	542,67	0,253	47	4,4	0,03
4	Parallel loops	69,80	1,969	366	7,8	0,24
5	Parallel divide and conquer	3,80	36,180	6 727	18,4	4,33
6	plus vectorization	1,10	124,914	23 224	3,5	14,96
7	plus AVX intrinsics	0,41	337,812	**62 806**	2,7	40,45

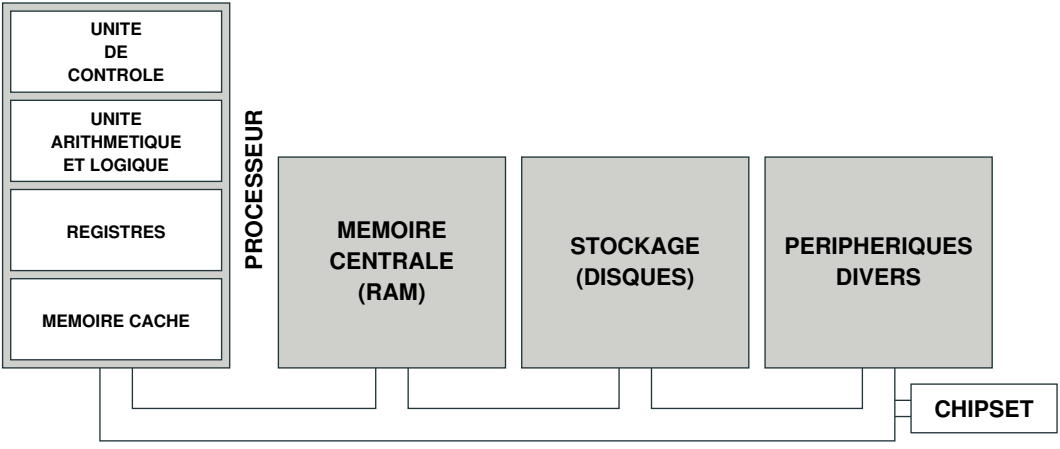
Top 3 du Top 500 : *benchmarks* HPL contre HPCG (11/2026)

Code	#	Machine	Country	Rpeak (Eflop/s)	Rmax (Eflop/s)	% RPeak	MW
HPL	1	El Capitan	USA	2,8	**1,8**	**64**	30
HPL	2	Frontier	USA	2	**1,3**	**65**	25
HPL	3	Aurora	USA	1,9	**1**	**53**	39
HPCG	1	El Capitan	USA	2,8	**0,017**	**0,6**	30
HPCG	2	Fugaku	Japan	0,4	**0,016**	**2,8**	30
HPCG	3	Frontier	USA	2	**0,014**	**0,7**	25

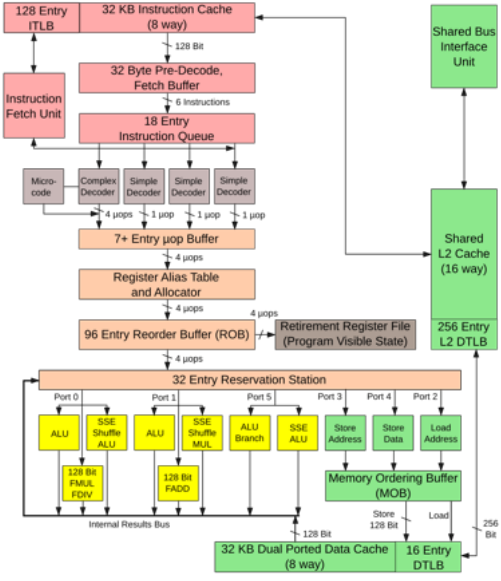
Source : Jack Dongarra, 2021 ACM A. M. Turing Award Lecture, SC 22 (Supercomputing 2022).
https://www.top500.org

Pourquoi est-il compliqué d'obtenir automatiquement du code efficace ?

Vue (très) simplifiée d'un ordinateur

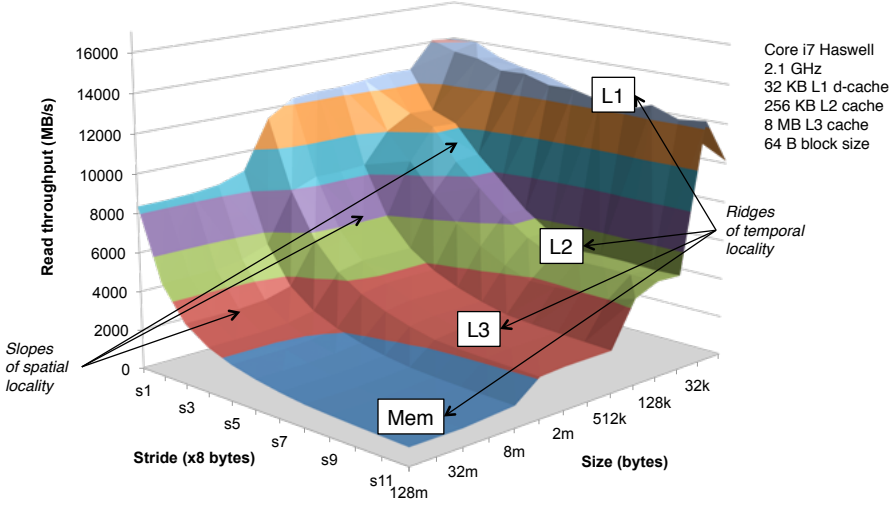


Vue (moins) simplifiée d'une architecture de processeur

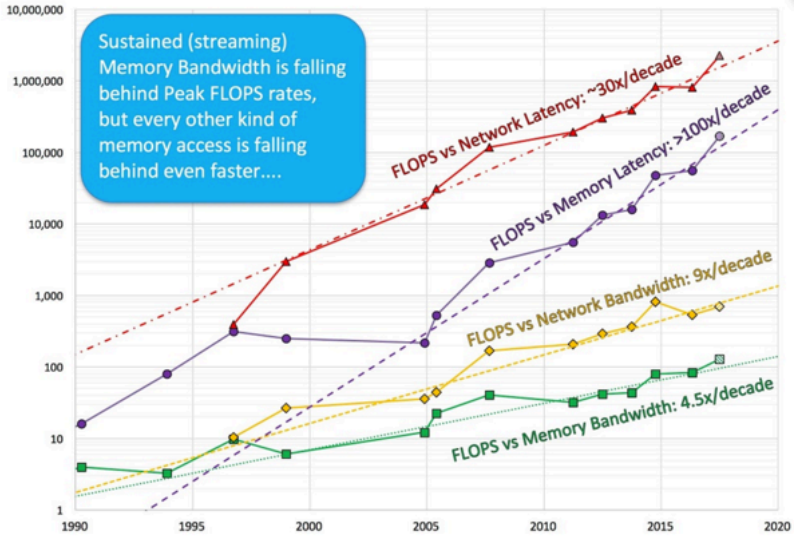


Intel Core 2 Architecture

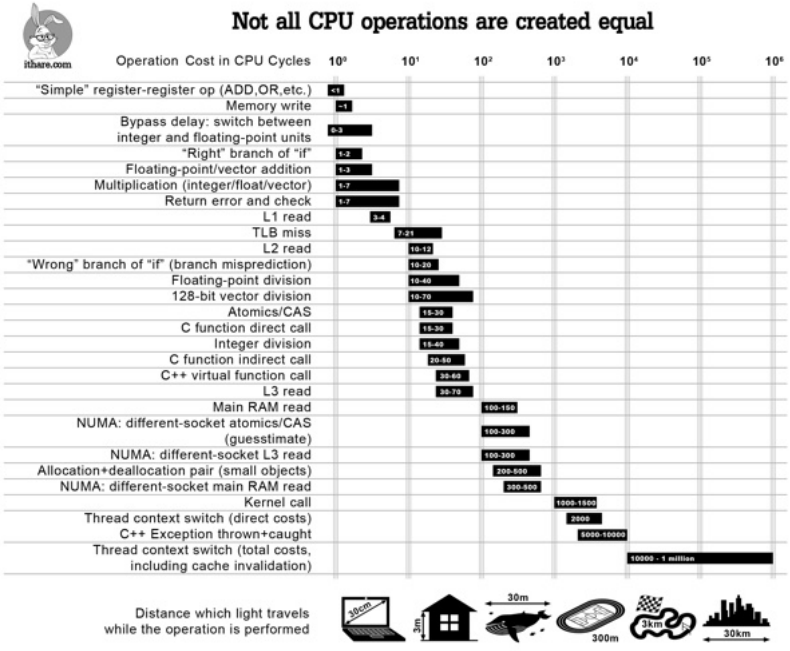
La montagne des débits mémoire



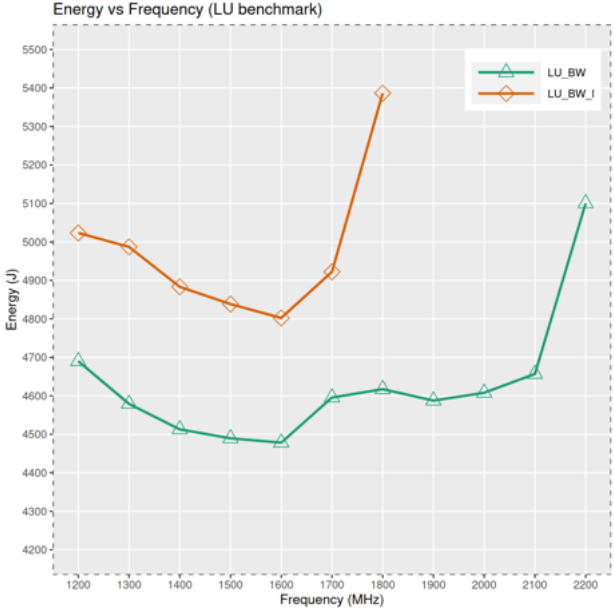
CPU vs mémoire et réseau



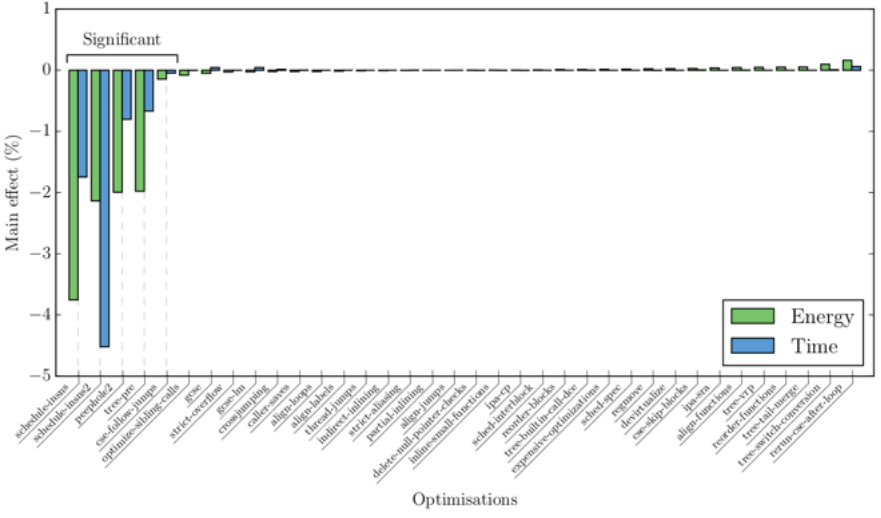
Coûts des opérations processeur



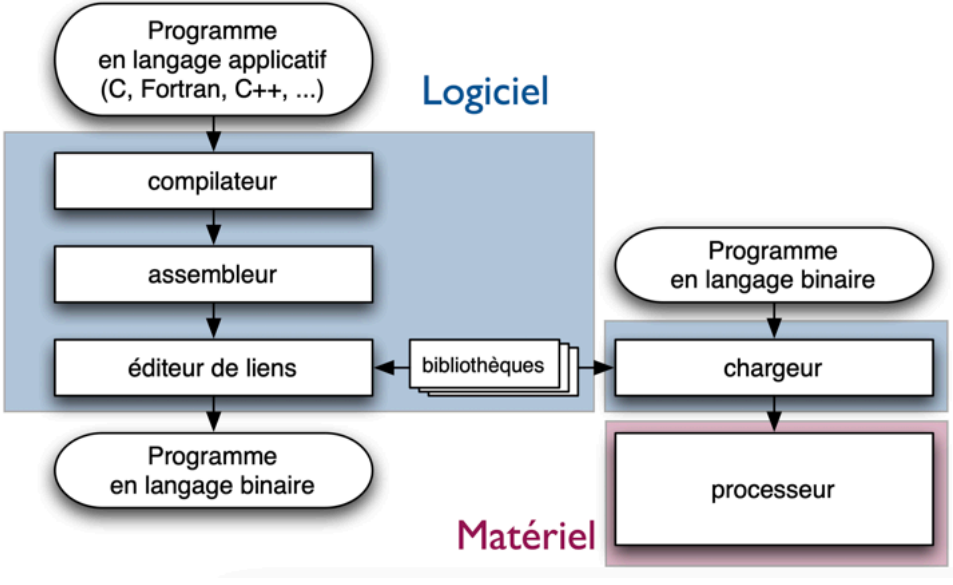
Relation entre fréquence du CPU et énergie consommée



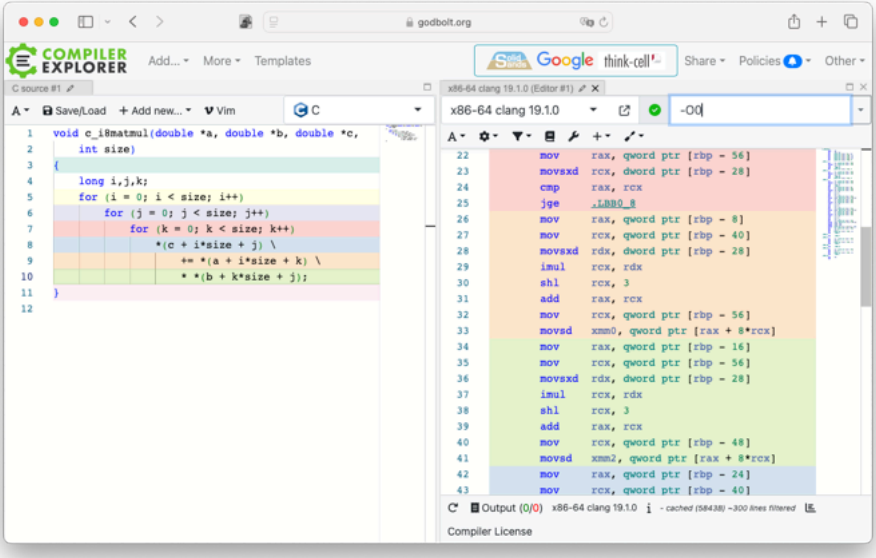
Optimiser pour l'énergie ne revient pas à optimiser pour le temps



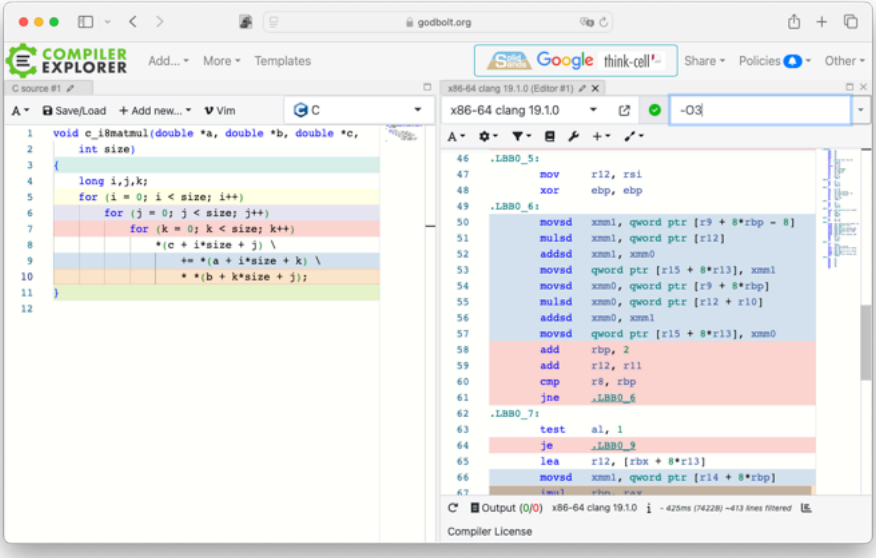
Chaîne de compilation et exécution



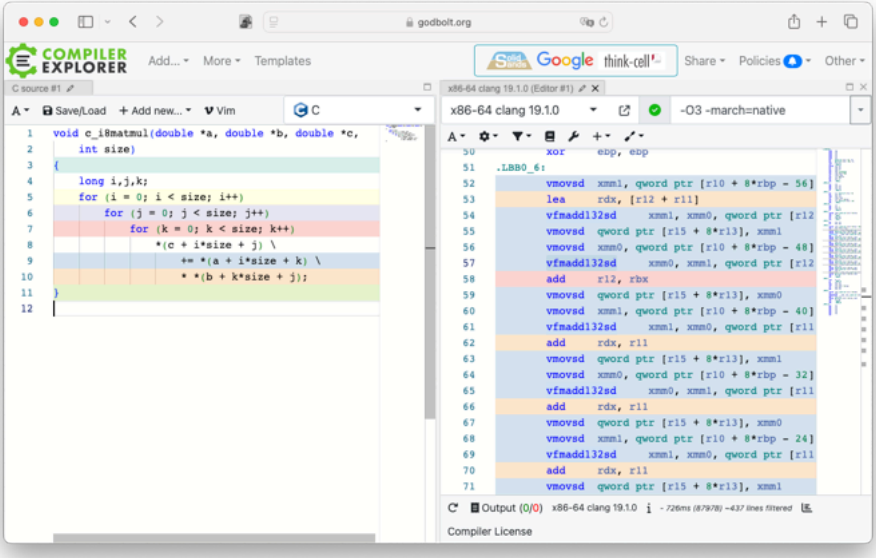
Compilation sans optimisation



Compilation avec optimisation -O3 (x86-64 ”classique”)



Compilation avec optimisation -O3 (x86-64 spécifique)



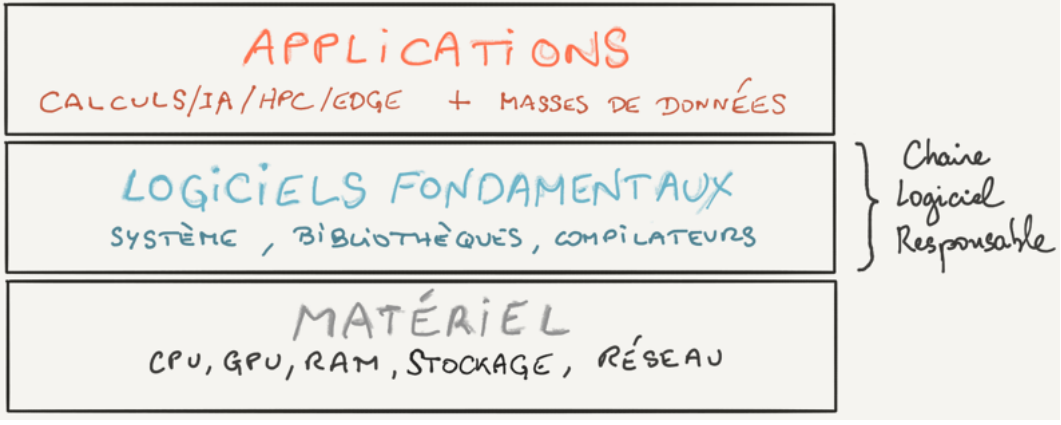
Une chaire sur le logiciel responsable

Quelques logiciels libres fondamentaux

L'**épine dorsale** de l'**infrastructure numérique** mondiale
est constituée de **logiciels libres**.

- Firmware et système d'exploitation (CoreBoot, Linux)
- Virtualisation (KVM, Kubernetes, Docker, OpenShift)
- Compilateurs optimisants (GCC, LLVM, rust)
- Bibliothèques (libc/libc++, libm, BLAS/LAPACK)
- Interpréteurs (Python, NodeJS, GraalVM, OpenJDK)
- Packages (numpy, pytorch, tensorflow, jax, xla, keras, Scikit-learn)
- Langages améliorés : Python, JavaScript, C, C++, Rust, Java, Fortran

Des logiciels libres fondamentaux plus efficaces et frugaux



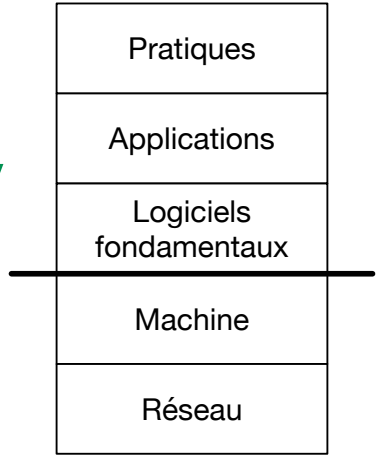
Améliorer les **logiciels libres** fondamentaux pour une **meilleure utilisation du matériel** par les **applications** : utiliser **moins d'énergie**, de **mémoire** et de **temps**, de manière plus **robuste** et plus **sûre**.

Deux approches complémentaires

Green IT

"vider sa boîte mail"
"regarder moins de vidéos"

éco-conception logicielle



Logiciel responsable

Optimiser / Prédire
temps, mémoire, énergie

Modéliser
Comprendre
Mesurer

Axes de recherche : faire mieux avec moins

Rendre les **logiciels libres fondamentaux** plus **frugaux**, plus **efficaces** et **plus sûrs**

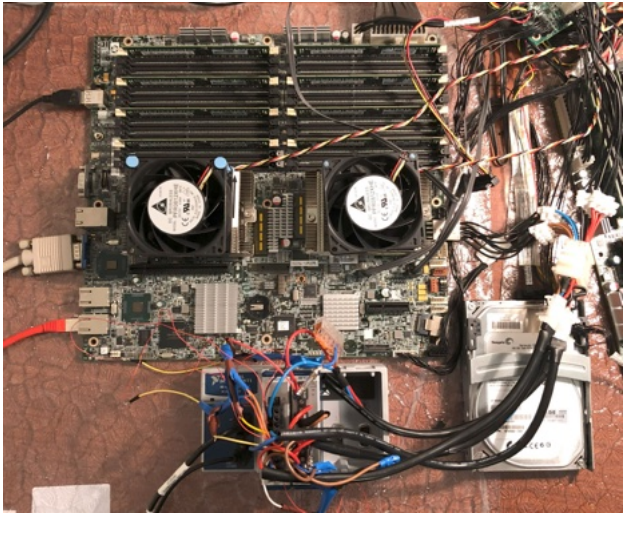
1. **Mesurer** et **comprendre** finement l'utilisation de l'énergie par le logiciel
2. Créer des **modèles de coûts** en énergie, temps, mémoire
3. Utiliser ces modèles dans les **compilateurs optimisants**
4. Générer **automatiquement** du code parallèle **frugal** et **efficace**
5. **Réduire** la **taille des données** et **la quantité de calculs**
6. Rendre visibles les coûts en **énergie**, **temps**, **mémoire**
7. **Stratégies** de **placement** de tâches **sensible à l'énergie**
8. **Définir** le **matériel minimal** pour une **tâche donnée**

https://mines.paris/lore

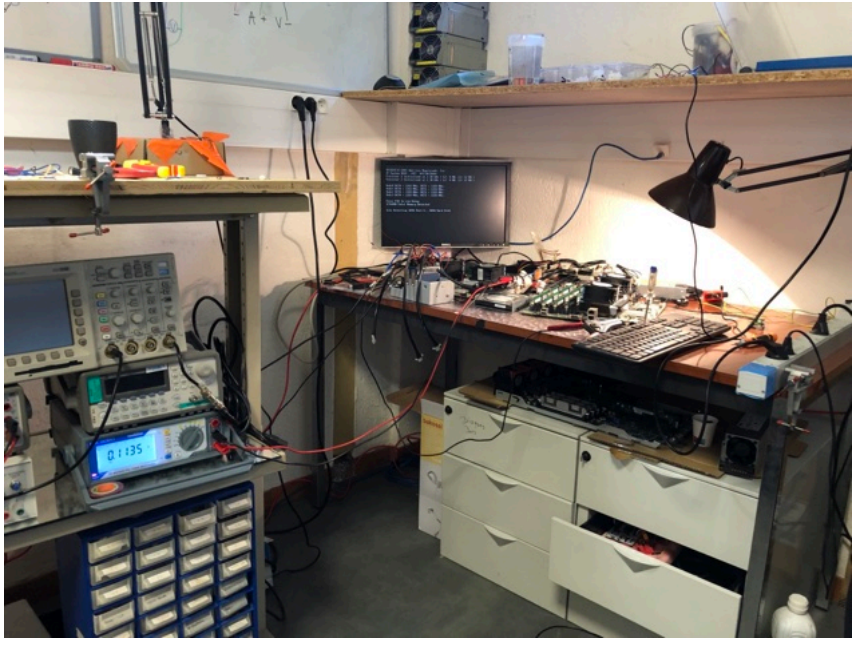
État de l'art partiel : focus sur l'énergie

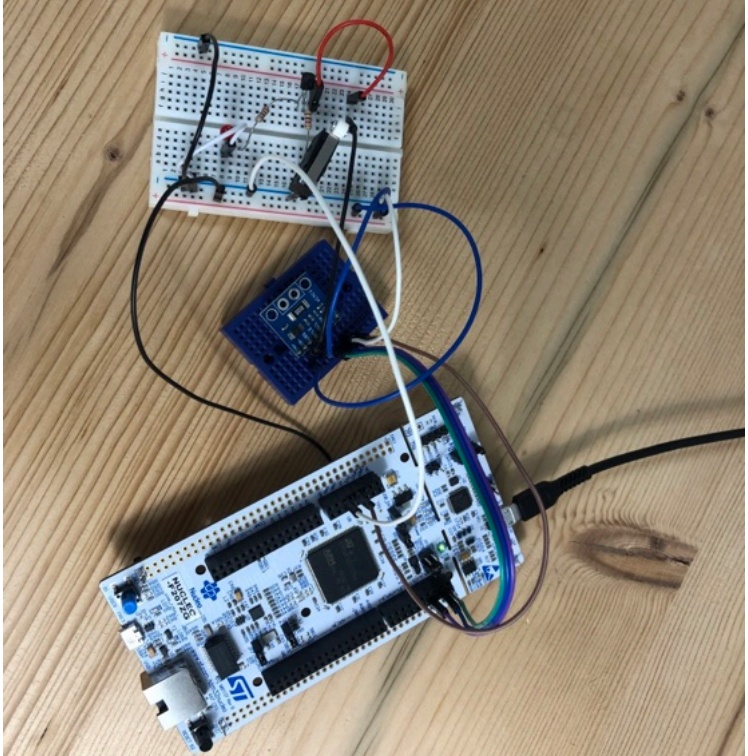
- James Pallister (2016), « *Exploring the fundamental differences between compiler optimisations for energy and for performance* » (doctorat).
- Thomas Ilsche (2020), « *Energy Measurements of High Performance Computing Systems : From Instrumentation to Analysis* » (doctorat).
- Anne-Cécile Orgerie (2020), « *From Understanding to Greening the Energy Consumption of Distributed Systems* » (HDR).
- Mathilde Jay (2024), « *A Versatile Methodology for Assessing the Electricity Consumption and Environmental Footprint of Machine Learning Training : from Supercomputers to Edge Devices* » (doctorat).

Axes 1/2 : thèse d'Antoine Solcourt (2023-2026)

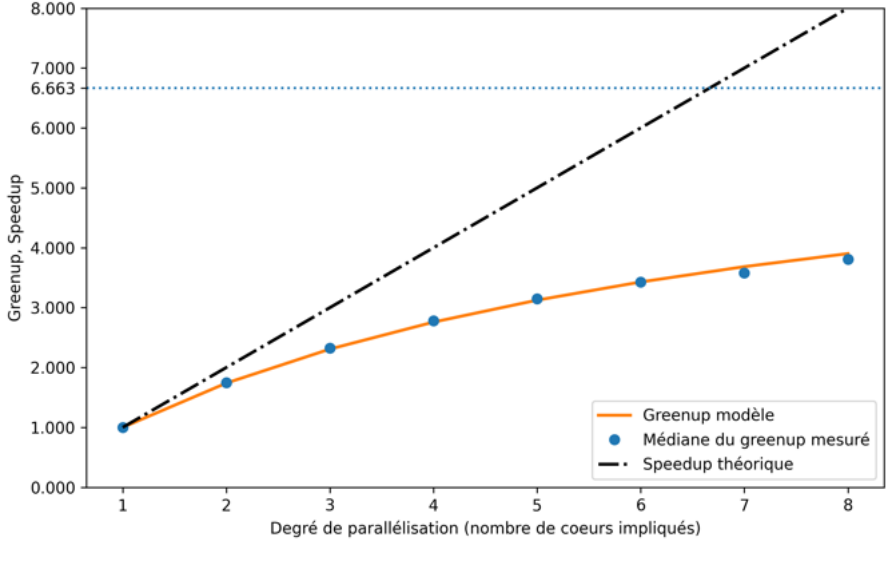


- Serveurs, desktops
- Conception de sondes de mesure tension/courant
- 25 points de mesure
- *Nano-benchmarks* maison
- *Bare metal*
- → système de mesure
- → modèles

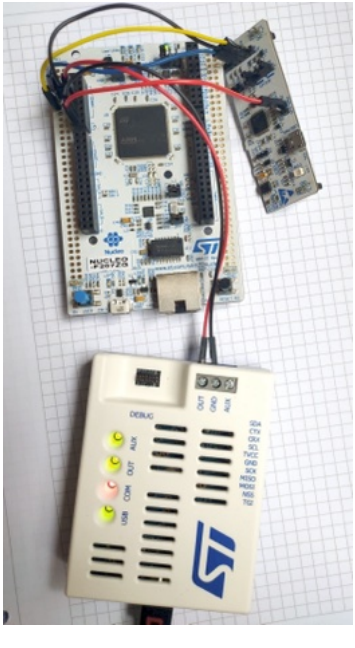




Borne supérieure du ”green-up” pour les codes parallèles



Axes 2/3/6 : thèse de Benjamin Destal (2023-2026)



- STM32 (SoC)
- Mesures tension/courant avec sondes ST
- *Benchmarks* maison
- → modèles
- → optimisations
- → visualisation

Modèle au niveau de l'instruction

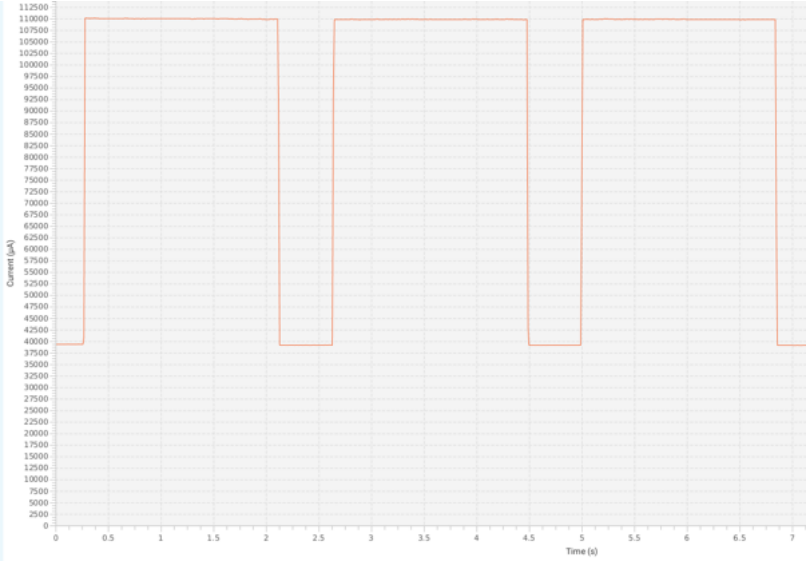
$$E_p = \sum_{i \in ISN} B_i N_i + \sum_{i,j} O_{i,j} N_{i,j} + \sum_k E_k$$

- B_i le coût énergétique de base de l'instruction i
- N_i le nombre d'occurrences de l'instruction i
- $O_{i,j}$ le coût «inter-instruction» de la paire d'instructions (i, j)
- $N_{i,j}$ le nombre d'occurrences de la paire d'instructions (i, j)
- E_k le surcoût énergétique lié notamment aux bulles dans le pipeline et aux défauts de cache

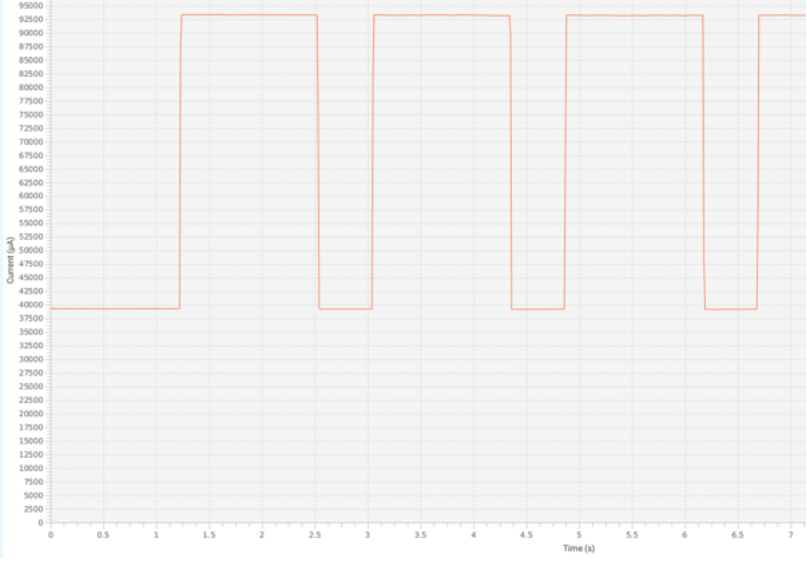
Microbenchmark

```
.microbenchmark:
    mov     r12, N  // Nombre d'itérations
.MICROBENCHMARK_LOOP:
    add.n   r0, r1, r2  // Motif: r0 <- r1 + r2 (taille_motif=1)
    add.n   r0, r1, r2  // Motif: r0 <- r1 + r2
    add.n   r0, r1, r2  // Motif: r0 <- r1 + r2
    (...) // répétitions du motif jusqu'à M fois (ici M=120)
    subs.w  r12, #1     // instruction de boucle
    bne.w     .MICROBENCHMARK_LOOP  // instruction de boucle
    bx      lr
```

Mesure 3M LOAD (ARM) avec cache hit



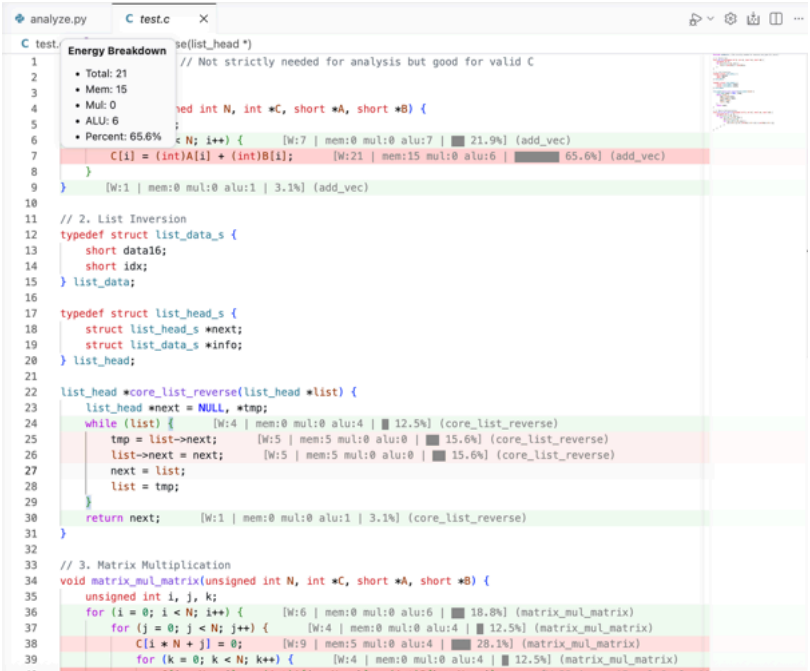
Mesure 300K LOAD (ARM) avec cache miss



Erreurs entre les prédictions des modèles et les mesures

Benchmark	Ordre 0	Ordre 1	Ordre 2
BB_1	-14.93 %	13.4 %	0.46 %
BB_2	0.27 %	14.72 %	0.55 %
BB_3	-14.15 %	29.11 %	1.45 %
BB_4	21.23 %	44.99 %	0.17 %
BB_5	-15.02 %	12.98 %	-0.75 %
BB_6	-1.30 %	13.4 %	0.3 %
BB_7	-10.81 %	10.67 %	-2.0 %

Restitution au niveau du code source



Portage par le Centre de recherche en informatique

- 30 ans d'**expérience** dans le domaine de l'**optimisation du logiciel**
- Contributions **scientifiques** majeures : *tiling*, interprocédural, énergie ...
- Nombreuses contributions **logicielles**, dont l'outil PIPS
- Transferts dans l'**industrie**, par des contrats et anciens élèves
- Contributions à des **compilateurs** industriels, dont GCC

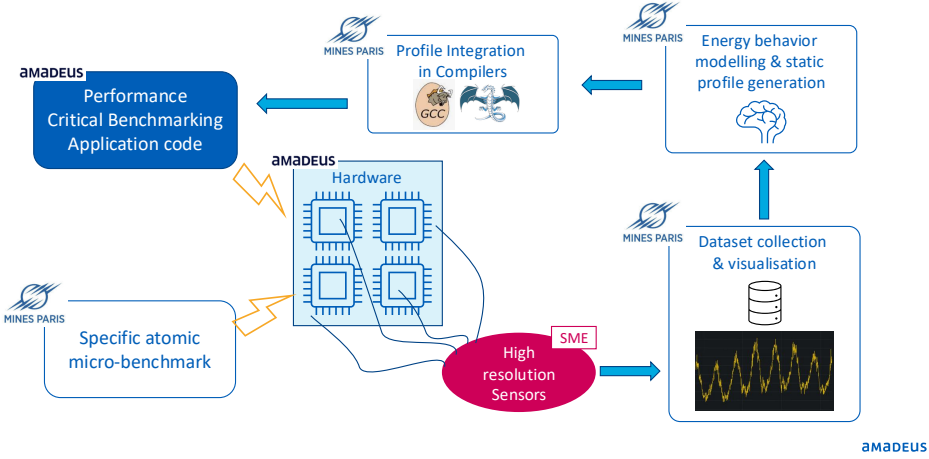
Collaborations industrielles en cours reliées à la chaire

Amadeus / **Électronie**. Projet PACE, financé par la BPI (2026–2030).
Compilation pour le temps et l'énergie de leur application principale de recherche (100 kTPS), guidée par les modèles obtenus par sondes de mesures matérielles.

ST Microelectronics. Thèse CIFRE en cours (Benjamin Destal).
Intégration de modèles de coûts dans l'IDE de développement de ST et dans le compilateur pour la plateforme STM32.
Prêt de matériel de laboratoire et de cartes STM32 pour la recherche et l'enseignement.

Energy guided optimisation for Compilers

A High Level representation



Visions 2030 : après les travaux de la chaire

- Matériel et protocole de **mesure d'énergie** pour obtenir des **modèles**
- Nouvelle option des **compilateurs** libres pour **optimiser l'énergie**
  `gcc -Oe` → code moins énergivore tout en restant performant
- Intégration aux environnements de développements logiciel de nouveaux outils pour **visualiser l'énergie consommée**
- Nouveaux mécanismes intégrés aux **systèmes d'exploitation** pour faire fonctionner les systèmes en **minimisant l'énergie**
- Métriques pertinentes pour décider **de ne pas faire** !
- Des élèves formés à **l'ingénierie logicielle** sur des **logiciels libres** de grande taille, pouvant promouvoir à leur tour **l'utilisation et la maintenance** de ces **communs numériques**
- Changement sociétal : un **usage prolongé du matériel** (> 10 ans ?)