

Logiciel responsable (LoRe)

Projet de chaire de recherche financée par le mécénat (2025–2030)

Georges-André Silber

<https://mines.paris/lore>

19 novembre 2024

Institut des transformations numériques (ITN)

Centre de recherche en informatique (CRI)

Mines Paris — PSL

Faire mieux avec moins

Responsabilité environnementale

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

Responsabilité sociétale

- contribuer aux logiciels libres, biens communs numériques
- produire des logiciels plus sûrs
- contribuer au financement de la recherche académique en ingénierie logicielle et à la formation

Contexte environnemental : un dépassement des limites planétaires

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).

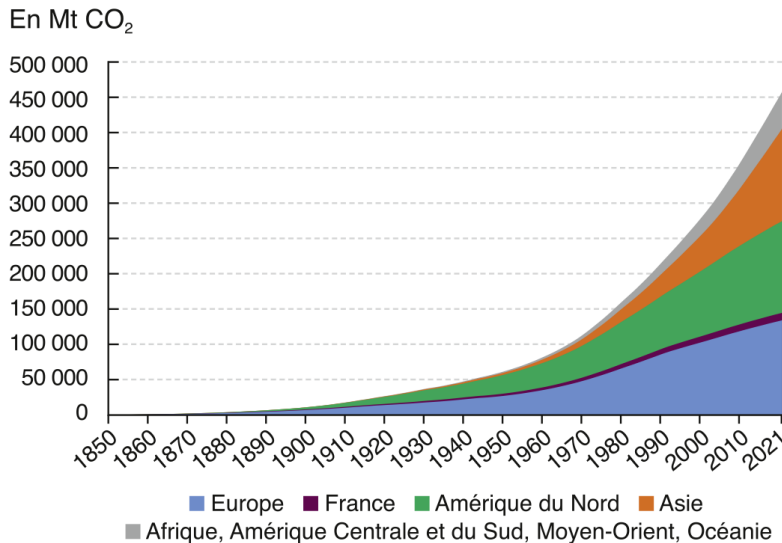
Les émissions

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 CO₂ évitée. « Le potentiel de réduction est de 31 gigatonnes à 2030 et 41 gigatonnes à 2035 », écrivent-ils.

Multiplier par six les investissements

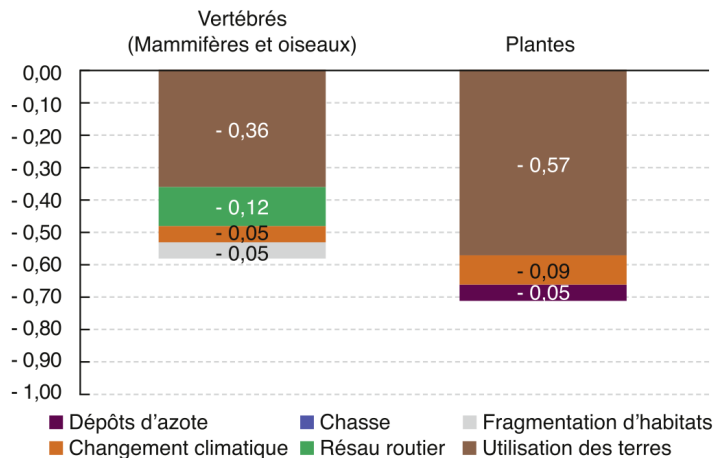
Et ce notamment grâce au développement d'électricité solaire photovoltaïque et éolienne, à des actions sur la forêt (meilleure gestion, arrêt de la déforestation, etc.), ou à d'autres mesures comme les économies d'énergie et une électrification

Concentration de CO₂ : lim. **350 ppm** → **379 ppm** (2005), **425 ppm** (2023)
↑ du forçage radiatif : lim. **+1,0 W/m²** → **+1,66 W/m²** (2005), **+2,72 W/m²** (2023)

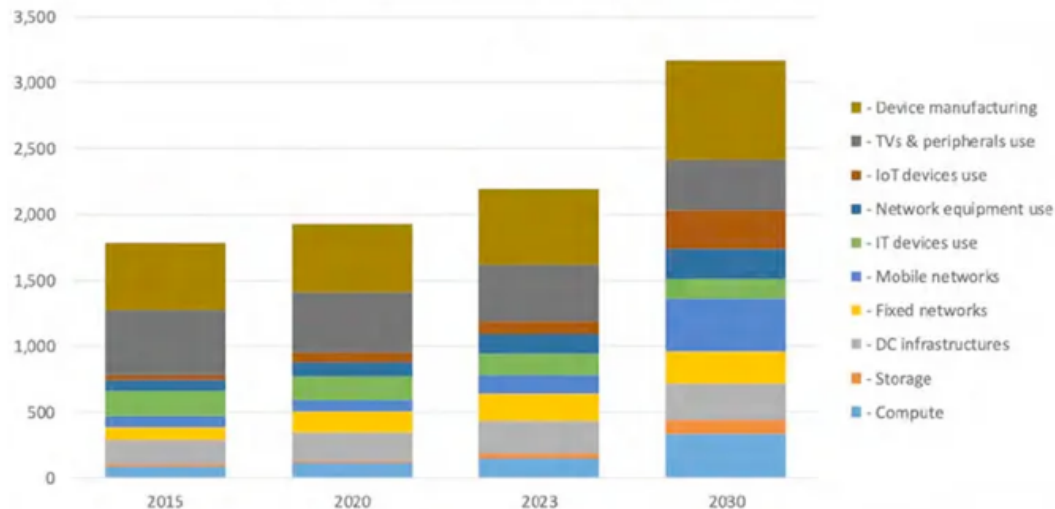


Source : Global Carbon Project, via <https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/la-france-face-aux-neuf-limites-planetaires/4-changement-climatique>.

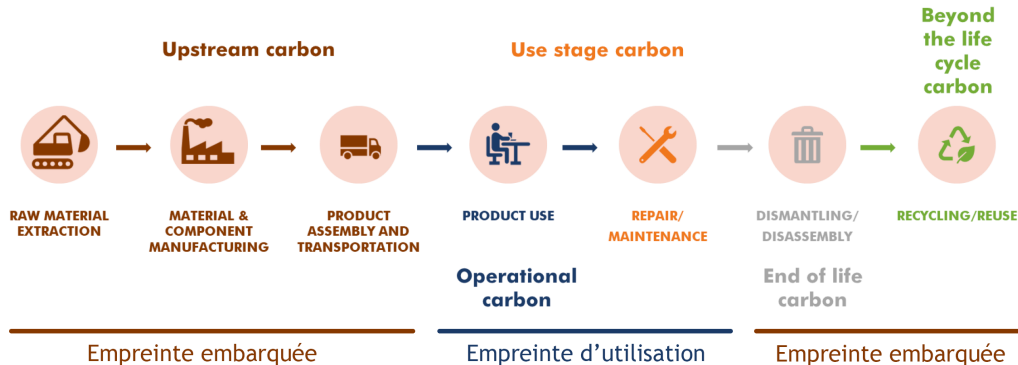
Pressions contribuant à la perte de l'abondance moyenne des espèces (MSA), en France métropolitaine, en 2020 (abondance moyenne perdue, 1-MSA)



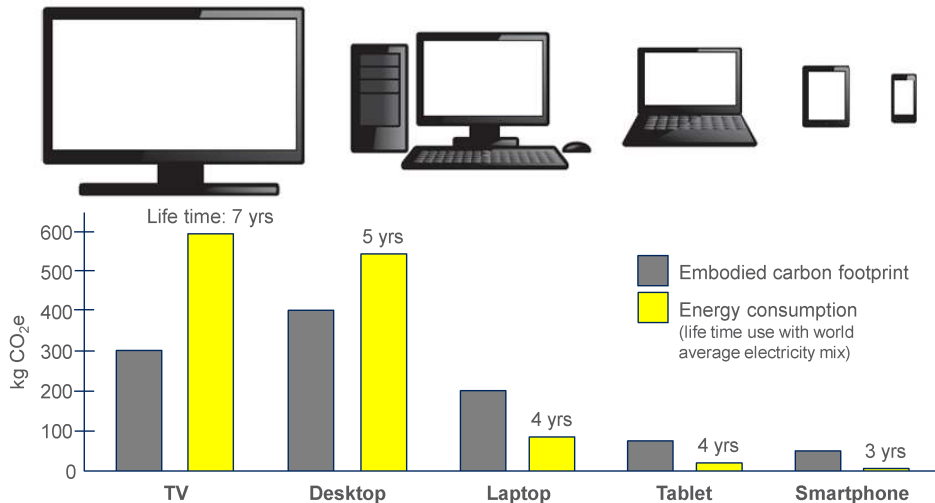
L'impact environnemental croissant du numérique



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



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



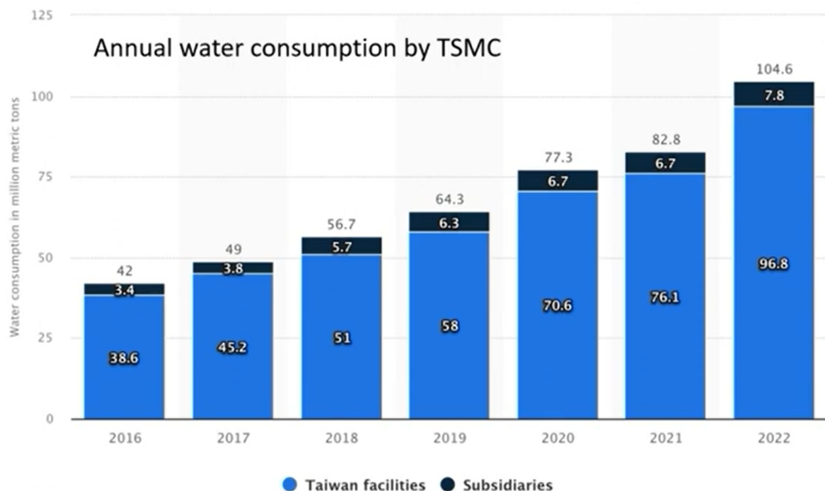
Source : J. Malmodin et D. Lundén, «The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010–2015», Sustainability, 9/2018.



Crédit : Arnout Fierens (<https://arnulf.be>).

I	II											III	IV	V	VI	VII	O
Hydrogène 1 H		1980's 1990's 2000's										2010's HgCdTe Excimer litho Radioactive					Helium 2 He
Lithium 3 Li	Béryllium 4 Be											Bore 5 B	Carbone 6 C	Azote 7 N	Oxygène 8 O	Fluor 9 F	Neon 10 Ne
Sodium 11 Na	Magnésium 12 Mg											Aluminium 13 Al	Silicium 14 Si	Phosphore 15 P	Soufre 16 S	Chlore 17 Cl	Argon 18 Ar
Potassium 19 K	Calcium 20 Ca	Scandium 21 Sc	Titane 22 Ti	Vanadium 23 V	Chromium 24 Cr	Manganèse 25 Mn	Fer 26 Fe	Cobalt 27 Co	Nickel 28 Ni	Cuivre 29 Cu	Zinc 30 Zn	Gallium 31 Ga	Germanium 32 Ge	Arsenic 33 As	Sélénium 34 Se	Bromure 35 Br	Krypton 36 Kr
Rubidium 37 Rb	Strontium 38 Sr	Yttrium 39 Y	Zirconium 40 Zr	Niobium 41 Nb	Molibdène 42 Mo	Tc 43	Ruthénium 44 Ru	Rhodium 45 Rh	Palladium 46 Pd	Argent 47 Ag	Cadmium 48 Cd	Indium 49 In	Étain 50 Sn	Antimoine 51 Sb	Tellure 52 Te	Iode 53 I	Xénon 54 Xe
Césium 55 Cs	Baryum 56 Ba	Lanthane 57 La	Hafnium 72 Hf	Ta 73	W 74	Re 75	Osmium 76 Os	Ir 77	Pt 78	Au 79	Hg 80	Tl 81	Pb 82	Bi 83	Po 84	At 85	Rn 86
Francium 87 Fr	Radium 88 Ra	Ce 58	Pr 59	Nd 60	Pm 61	Sm 62	Eu 63	Gd 64	Tb 65	Dy 66	Ho 67	Er 68	Tm 69	Yb 70	Lu 71		
		Th 90	Pa 91	U 92	Np 93	Pu 94	Am 95	Cm 96	Bk 97	Cf 98	Es 99	Fm 100	Md 101	No 102	Lr 103		

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



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



- 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?
- Chaire LoRe : Q2 et Q3
- → Utilisation pour nos mesures

Contexte technologique : *bloatware*

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>

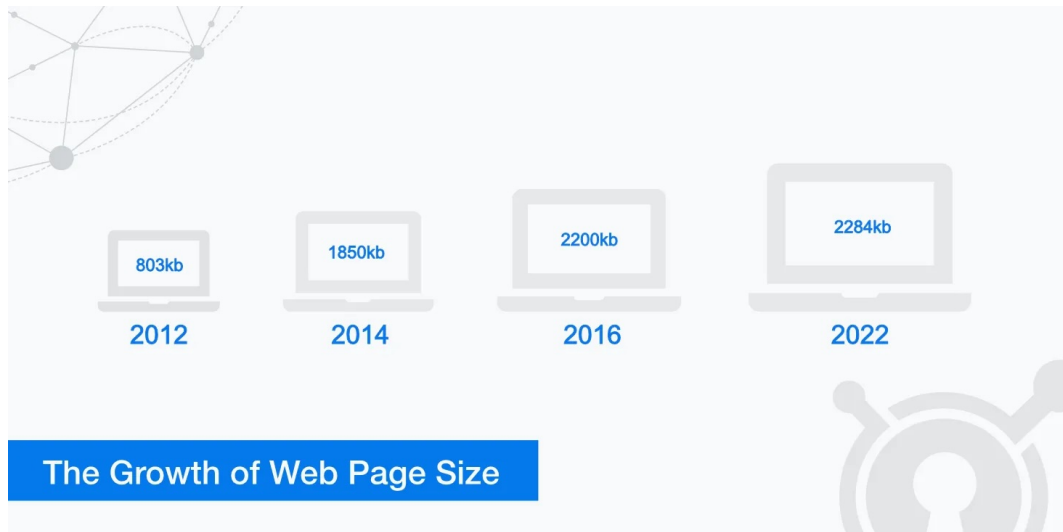
Le logiciel est comme un gaz, il se répand autant que le permet son support. Corollaire, le logiciel se développe jusqu'à ce qu'il soit limité par la loi de Moore. Il n'atteindra jamais un stade de maturité industrielle. L'industrie du logiciel est et restera toujours en état de crise.

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=gE4N0G9kE1M>

chaire_mines_lore.tex : 22 126 octets $\approx$ 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



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)

- 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.

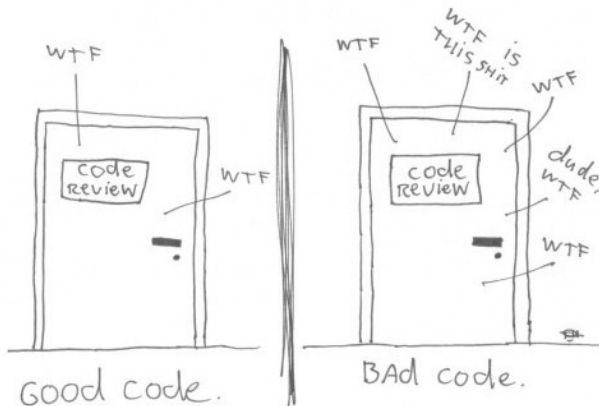
- L'éducation d'un programmeur : écriture de petits programmes
- Peu de préparation à la création de logiciels à grande échelle
- Mythe du héros, épreuves de codage pour l'embauche, hackathons
- Peu de réponse à : « *qu'est-ce qu'un bon logiciel ?* »
- Un diplôme en informatique ne garantit pas un savoir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

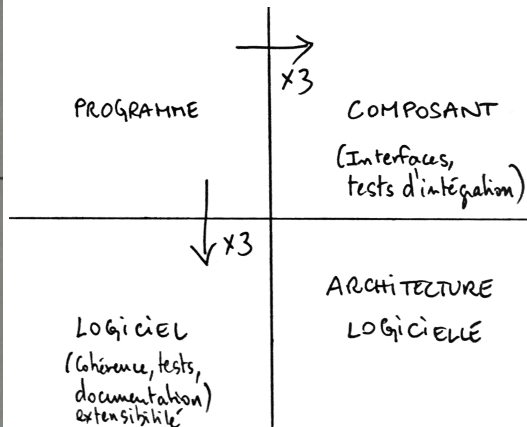
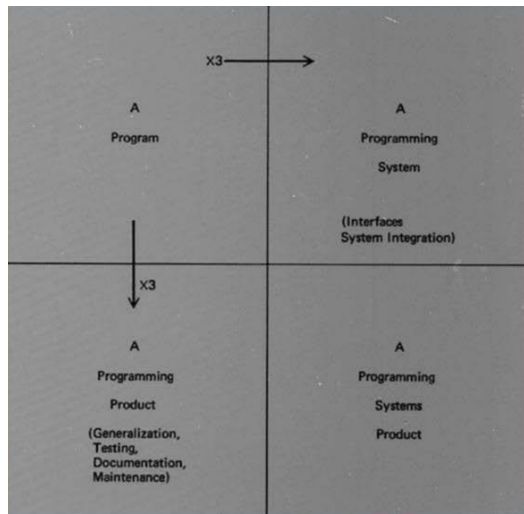
L'ingénierie logicielle est victime de cette fausse idée :

« *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](#) »

The ONLY VALID MEASUREMENT
OF CODE QUALITY: WTFs/MINUTE





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

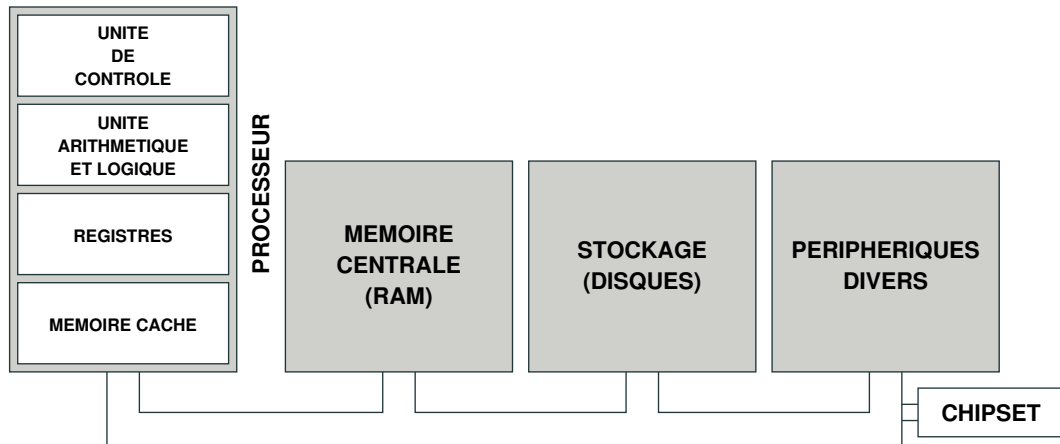
```
# Exemple: multiplication de matrices
for i in range(4096):
    for j in range(4096):
        for k in range(4096):
            C[i][j] += A[i][k] * B[k][j]
```

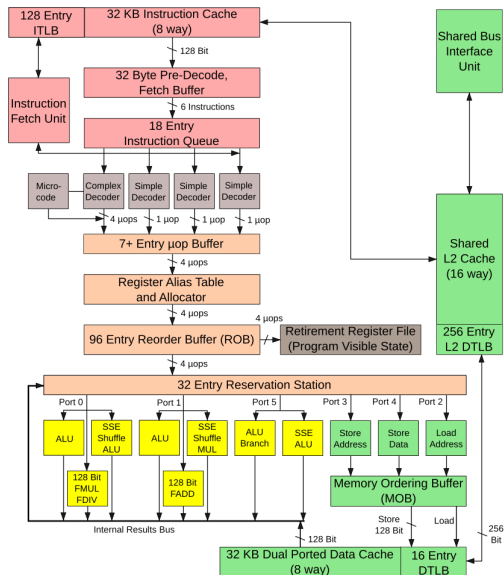
#	Method	Time (s)	Gflop/s	Speedup	Rel. sp.	% peak
1	Python	25 552,48	0,005	1	—	0,00
2	Java	2 372,68	0,058	11	10,8	0,01
3	C	542,67	0,253	47	4,4	0,03
4	Parallel loops	69,80	1,969	366	7,8	0,24
5	Parallel divide and conquer	3,80	36,180	6 727	18,4	4,33
6	plus vectorization	1,10	124,914	23 224	3,5	14,96
7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

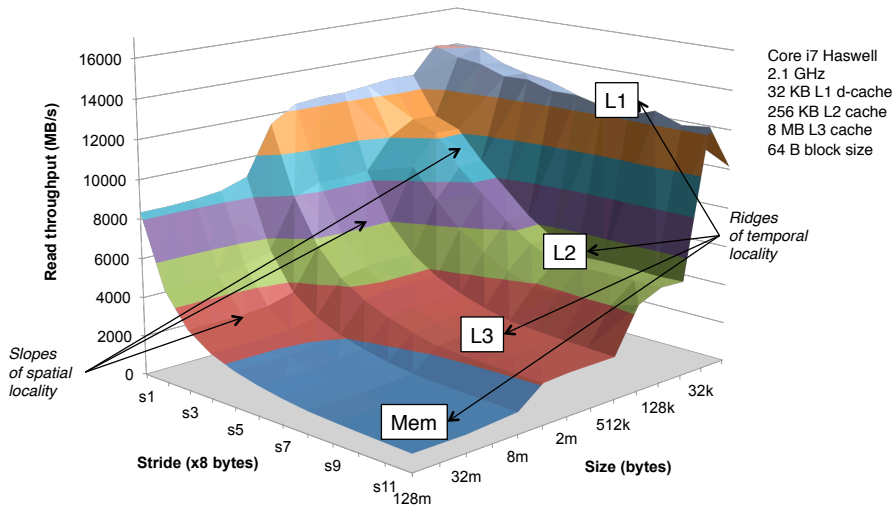
Code	#	Machine	Country	Rpeak (Tflop/s)	Rmax (Tflop/s)	% Rpeak	MW	Eff. (Gflop/s/W)
HPL	1	Frontier	USA	1685	1102	65	21.1	52.23
HPL	2	Fugaku	Japan	537	442	82	29.9	14.78
HPL	3	LUMI	Finland	428	309	72	6.0	51.38
HPCG	1	Fugaku	Japan	537	16	3,0	29.9	0.54
HPCG	2	Frontier	USA	1685	14,05	0,8	21.1	0.67
HPCG	3	LUMI	Finland	428	3,41	0,8	6.0	0.57

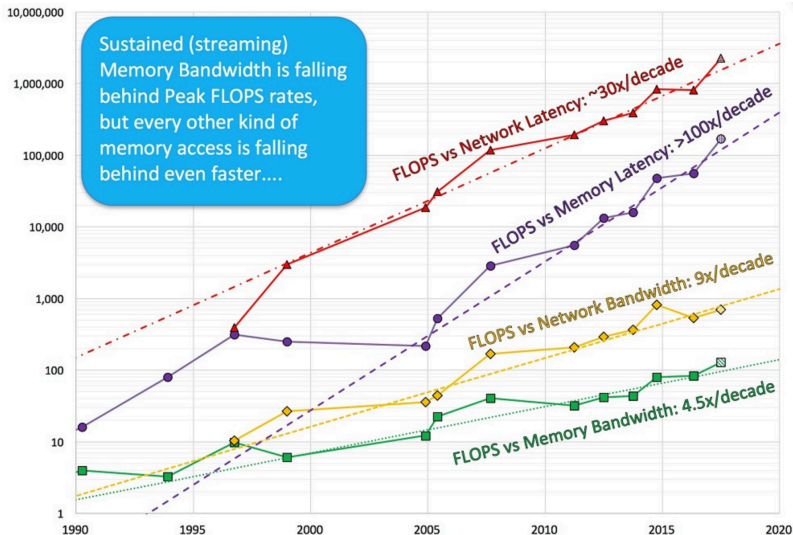
Source : Jack Dongarra, 2021 ACM A. M. Turing Award Lecture, SC 22 (Supercomputing 2022).

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



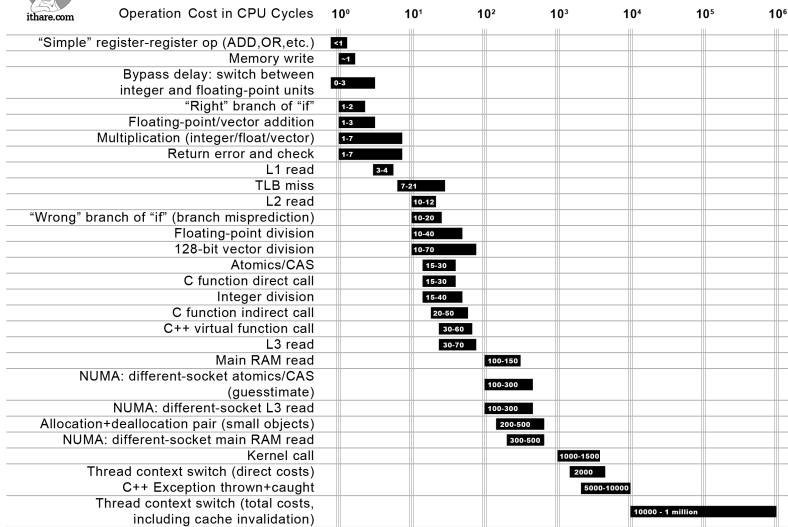






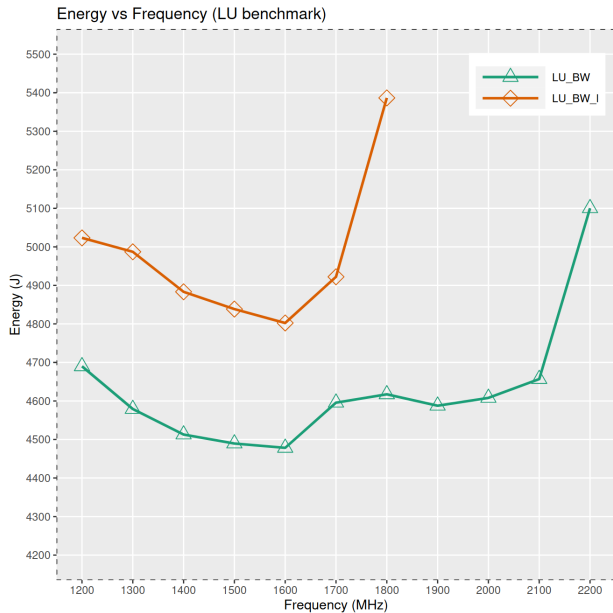


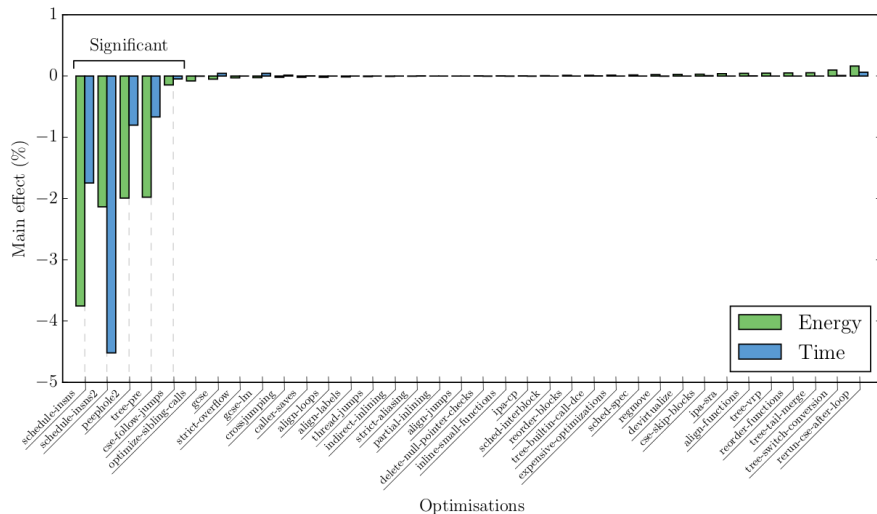
Not all CPU operations are created equal

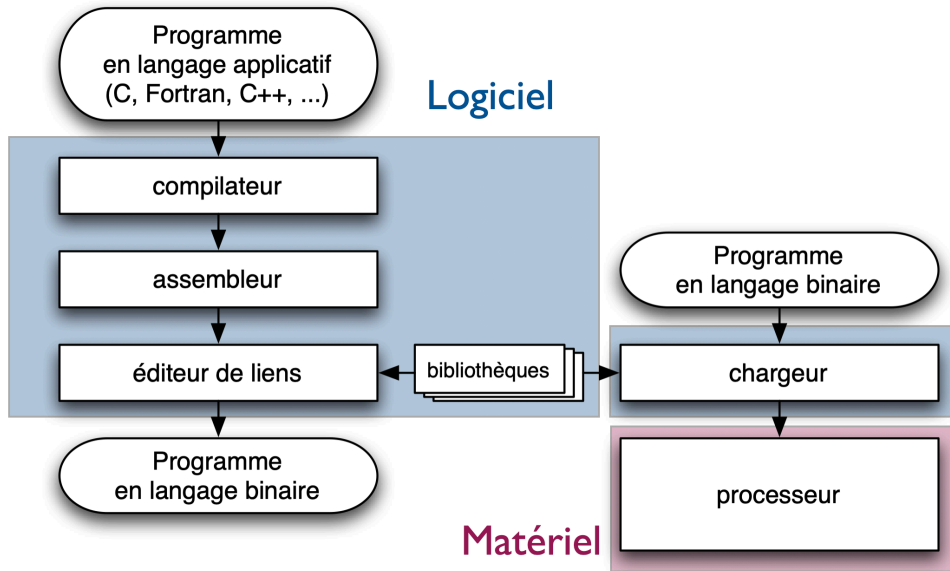


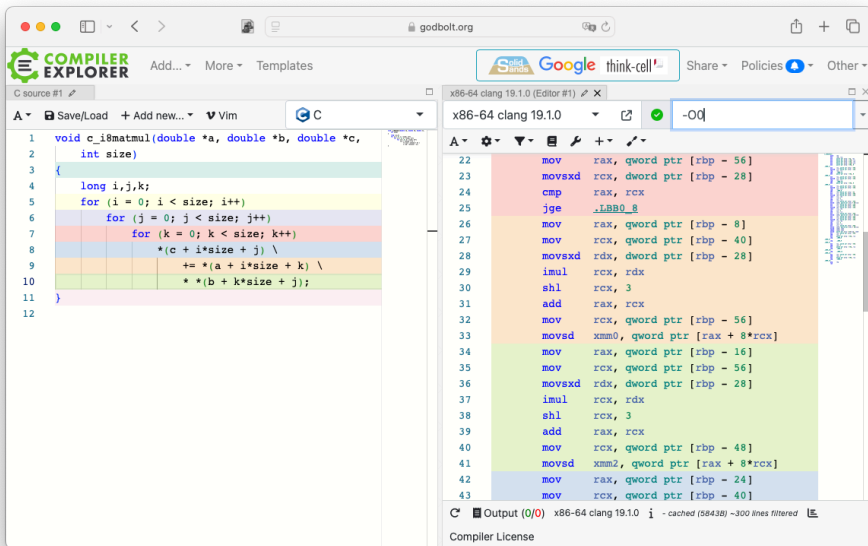
Distance which light travels while the operation is performed











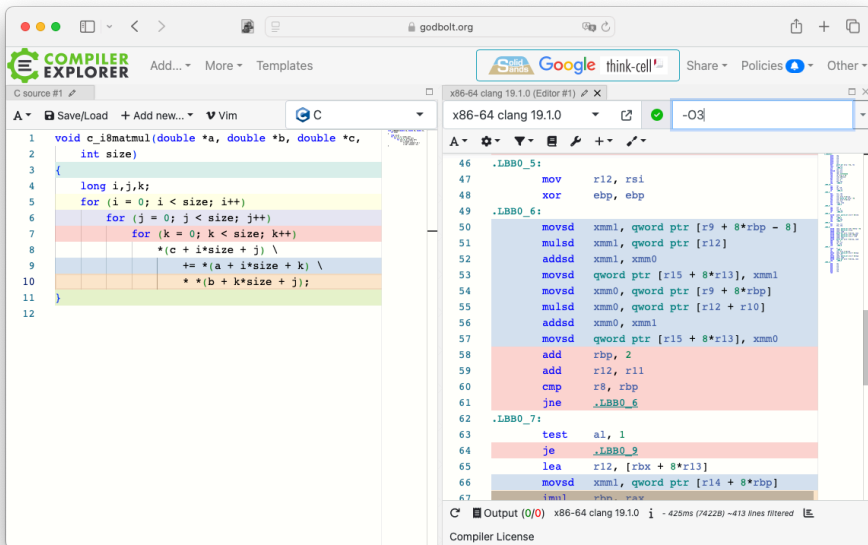
The screenshot shows the Compiler Explorer interface. The left pane displays the C source code for a function `c_i8matmul` that performs a matrix multiplication of three 2D arrays of type `double`. The right pane shows the assembly output generated by `x86-64 clang 19.1.0` with the `-O0` optimization level. The assembly code is color-coded to match the source code blocks: initialization and loop setup (pink), inner loop body (orange), and memory access/accumulation (green and blue).

```
1 void c_i8matmul(double *a, double *b, double *c,  
2 int size)  
3 {  
4     long i,j,k;  
5     for (i = 0; i < size; i++)  
6         for (j = 0; j < size; j++)  
7             for (k = 0; k < size; k++)  
8                 *(c + i*size + j) \  
9                 += *(a + i*size + k) \  
10                * *(b + k*size + j);  
11 }  
12
```

```
22     mov     rax, qword ptr [rbp - 56]  
23     movsxd  rcx, dword ptr [rbp - 28]  
24     cmp     rax, rcx  
25     jge     .LBB0_8  
26     mov     rax, qword ptr [rbp - 8]  
27     mov     rcx, qword ptr [rbp - 40]  
28     movsxd  rdx, dword ptr [rbp - 28]  
29     imul    rcx, rdx  
30     shl     rcx, 3  
31     add     rax, rcx  
32     mov     rcx, qword ptr [rbp - 56]  
33     movsd   xmm0, qword ptr [rax + 8*rcx]  
34     mov     rax, qword ptr [rbp - 16]  
35     mov     rcx, qword ptr [rbp - 56]  
36     movsxd  rdx, dword ptr [rbp - 28]  
37     imul    rcx, rdx  
38     shl     rcx, 3  
39     add     rax, rcx  
40     mov     rcx, qword ptr [rbp - 48]  
41     movsd   xmm2, qword ptr [rax + 8*rcx]  
42     mov     rax, qword ptr [rbp - 24]  
43     mov     rcx, qword ptr [rbp - 40]
```

Output (0/0) x86-64 clang 19.1.0 i - cached (5843B) ~300 lines filtered

Compiler License

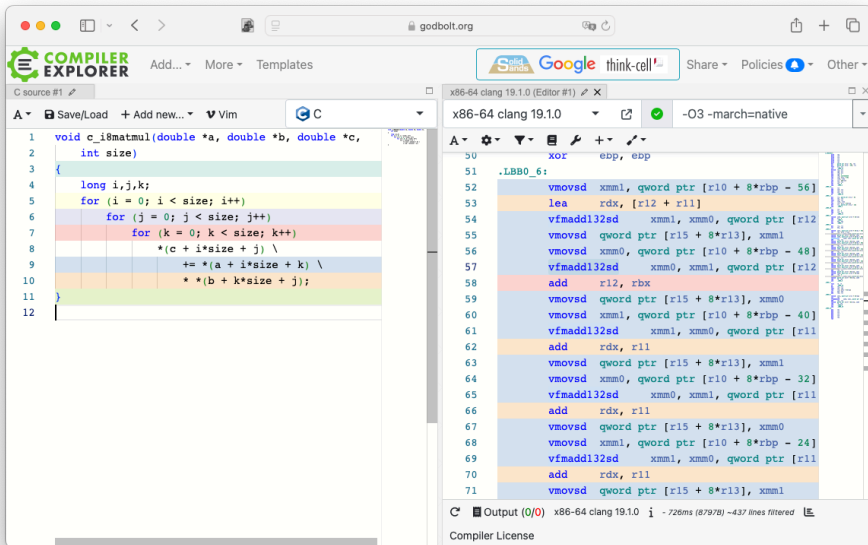


The screenshot shows the Compiler Explorer interface with the following details:

- Language:** C
- Compiler:** x86-64 clang 19.1.0
- Optimization Level:** -O3
- Source Code (C):**

```
1 void c_i0matmul(double *a, double *b, double *c,  
2 int size)  
3 {  
4     long i,j,k;  
5     for (i = 0; i < size; i++)  
6         for (j = 0; j < size; j++)  
7             for (k = 0; k < size; k++)  
8                 *(c + i*size + j) \  
9                 += *(a + i*size + k) \  
10                * *(b + k*size + j);  
11 }  
12
```
- Assembly Output (x86-64):**

```
46 .LBB0_5:  
47     mov     r12, rsi  
48     xor     ebp, ebp  
49 .LBB0_6:  
50     movsd   xmm1, qword ptr [r9 + 8*rbp - 8]  
51     mulsd   xmm1, qword ptr [r12]  
52     addsd   xmm1, xmm0  
53     movsd   qword ptr [r15 + 8*r13], xmm1  
54     movsd   xmm0, qword ptr [r9 + 8*rbp]  
55     mulsd   xmm0, qword ptr [r12 + r10]  
56     addsd   xmm0, xmm1  
57     movsd   qword ptr [r15 + 8*r13], xmm0  
58     add     rbp, 2  
59     add     r12, r11  
60     cmp     r8, rbp  
61     jne     .LBB0_6  
62 .LBB0_7:  
63     test    al, 1  
64     je      .LBB0_9  
65     lea     r12, [rbx + 8*r13]  
66     movsd   xmm1, qword ptr [r14 + 8*rbp]  
67     imul    rbp, rax
```
- Output:** x86-64 clang 19.1.0 | - 425ms (74228) ~413 lines filtered
- Footer:** Compiler License



The screenshot shows the Compiler Explorer interface. The left pane displays the C source code for a matrix multiplication function. The right pane shows the generated assembly code for x86-64 using clang 19.1.0 with the -O3 -march=native optimization level. The assembly code includes instructions for memory access, vector register manipulation (xmm), and arithmetic operations.

```
1 void c_i8matmul(double *a, double *b, double *c,  
2 int size)  
3 {  
4     long i,j,k;  
5     for (i = 0; i < size; i++)  
6         for (j = 0; j < size; j++)  
7             for (k = 0; k < size; k++)  
8                 *(c + i*size + j) \  
9                 += *(a + i*size + k) \  
10                *(b + k*size + j);  
11 }  
12
```

```
50 xor     ebp, ebp  
51 .LBB0_6:  
52 vmovsd  xmm1, qword ptr [r10 + 8*rbp - 56]  
53 lea     rdx, [r12 + r11]  
54 vfmadd132sd  xmm1, xmm0, qword ptr [r12]  
55 vmovsd  qword ptr [r15 + 8*r13], xmm1  
56 vmovsd  xmm0, qword ptr [r10 + 8*rbp - 48]  
57 vfmadd132sd  xmm0, xmm1, qword ptr [r12]  
58 add     r12, rbx  
59 vmovsd  qword ptr [r15 + 8*r13], xmm0  
60 vmovsd  xmm1, qword ptr [r10 + 8*rbp - 40]  
61 vfmadd132sd  xmm1, xmm0, qword ptr [r11]  
62 add     rdx, r11  
63 vmovsd  qword ptr [r15 + 8*r13], xmm1  
64 vmovsd  xmm0, qword ptr [r10 + 8*rbp - 32]  
65 vfmadd132sd  xmm0, xmm1, qword ptr [r11]  
66 add     rdx, r11  
67 vmovsd  qword ptr [r15 + 8*r13], xmm0  
68 vmovsd  xmm1, qword ptr [r10 + 8*rbp - 24]  
69 vfmadd132sd  xmm1, xmm0, qword ptr [r11]  
70 add     rdx, r11  
71 vmovsd  qword ptr [r15 + 8*r13], xmm1
```

Output (0/0) x86-64 clang 19.1.0 i - 726ms (87978) -437 lines filtered

Compiler License

Une chaire sur le logiciel responsable

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

APPLICATIONS

CALCULS/IA/HPC/EDGE + MASSES DE DONNÉES

LOGICIELS FONDAMENTAUX

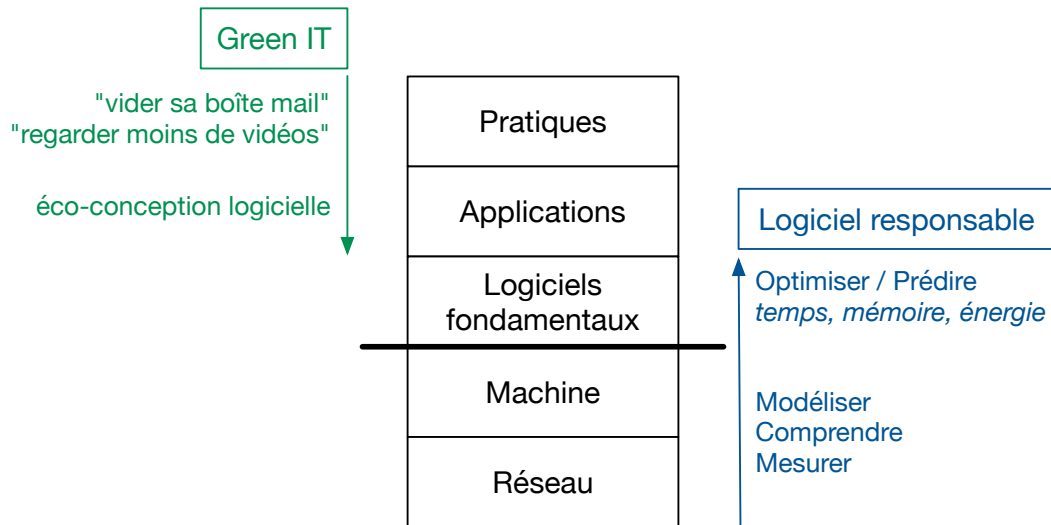
SYSTÈME, BIBLIOTHÈQUES, COMPILATEURS

MATÉRIEL

CPU, GPU, RAM, STOCKAGE, RÉSEAU

} Chaire
Logiciel
Responsable

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**.

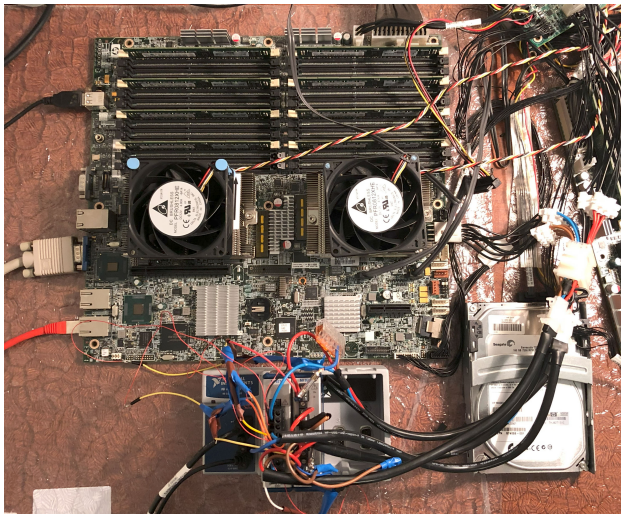


Rendre les **logiciels libres fondamentaux** plus **frugaux**, plus **efficaces** et **plus sûrs**

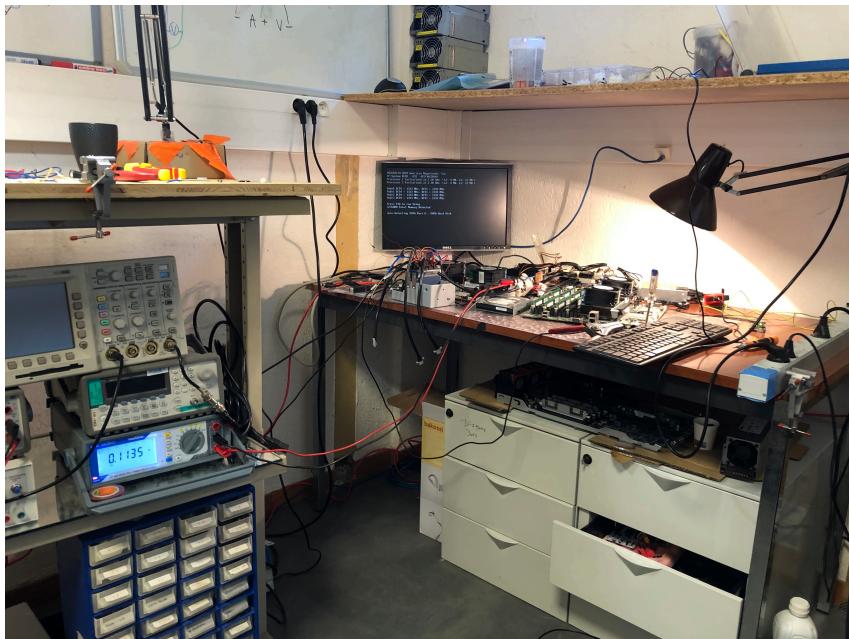
1. **Mesurer** et **comprendre** finement l'utilisation de l'énergie par le logiciel
2. Créer des **modèles de coûts** en énergie, temps, mémoire
3. Utiliser ces modèles dans les **compilateurs optimisants**
4. Générer **automatiquement** du code parallèle **frugal** et **efficace**
5. **Réduire** la **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**

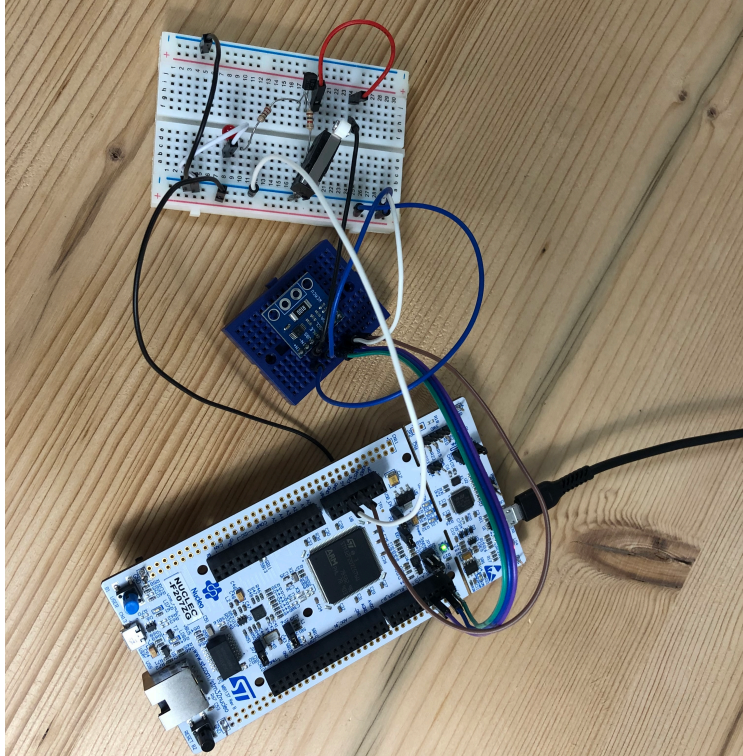
<https://mines.paris/lore>

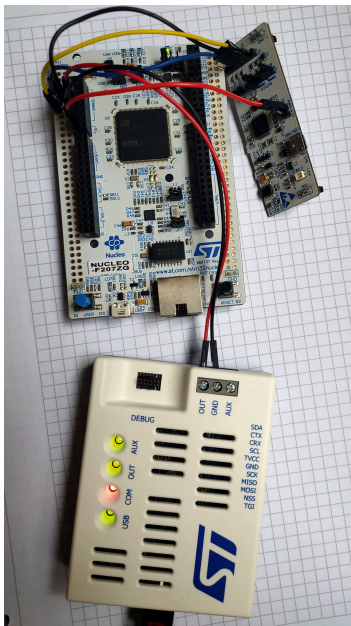
- James Pallister (2016), « *Exploring the fundamental differences between compiler optimisations for energy and for performance* » (doctorat).
- Thomas Ilsche (2020), « *Energy Measurements of High Performance Computing Systems : From Instrumentation to Analysis* » (doctorat).
- Anne-Cécile Orgerie (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).



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







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

- 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



- Une équipe du CRI à Sophia dédiée à la chaire :
 - 1 **enseignant-chercheur** permanent responsable (GAS)
 - 1 **ingénieur de recherche** permanent (Laurent Daverio)
 - 2 **doctorants** (Antoine Solcourt, Benjamin Destal)
 - 1 **stagiaire ingénieur** (François Duhil de Benazé)
- Les **enseignants-chercheurs** du CRI à Fontainebleau participant à la chaire :
Corinne Ancourt, Pierre Jouvelot, Fabien Coelho, Olivier Hermant, Claude Tadonki
- 230 **enseignants-chercheurs** pluri-disciplinaires des Mines en appui

- **Stages ingénieurs** de 6 mois
- **Thèses** de trois ans
- **Post-docs** de deux ans
- **Ingénieurs de recherche** sur 5 ans
- **Matériel** de mesure et de développement logiciel
- **Diffusion des connaissances** (publications, conférences, communication)
- **Coûts d'environnement** (gestion, missions, locaux)
- **Adhésion à l'ITN** de l'entreprise mécène (club des partenaires)

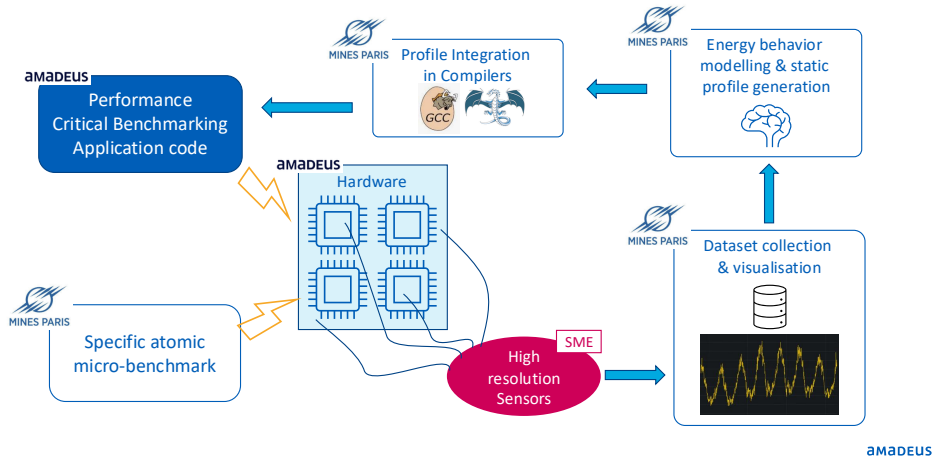
- Sujets de **travaux pratiques** et de **projets** pour les élèves ingénieurs de l'**option informatique** de 3A des mines
- **Projets d'ingénierie** en 2A des mines : **informatique soutenable**
- **CodeMineur** : stages rémunérés de 6 mois pour les élèves. Modèle : [Google Summer of Code](#). **Développer** un point particulier dans un **logiciel libre fondamental**. Sujets définis avec les entreprises participant à la chaire
- **Séminaires, tutoriels** en ligne, organisation de **formations** auprès des entreprises adhérentes

Amadeus / Électronie. Projet PACE en cours de définition (2025–2029).
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



- 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?)

Modalités

- **Urgence environnementale**

« optimiser sous contraintes dans un monde fini ».

- **Améliorer** des **logiciels que vous utilisez** directement ou indirectement
- **Contribuer** au financement de **biens communs fondamentaux**
- **Prolonger les travaux** par des **collaborations contractuelles**
- Renforcer la **formation** des ingénieurs logiciels, les **recruter** plus facilement
- Renforcer la **marque** de votre entreprise par un mécénat d'**intérêt public**

- **Partenaires recherchés** : entreprises produisant du logiciel ou utilisant des logiciels fondamentaux open source
- **Modalités d'adhésion** : cotisation modulable de 100 k€ par an (réduction d'impôts de 60 %)
- **Objectif de financement** : 5 partenaires. Budget total de 500 k€/an, soit 2,5 M€ pour 5 ans
- **Gouvernance** : programme scientifique, comité d'orientation incluant les entreprises mécènes. Rapport scientifique annuel
- **Dissémination** : contributions aux logiciels libres. Publications dans des journaux et conférences. Séminaires et workshops ouverts aux mécènes
- **Contact** : Georges-André Silber
georges-andre.silber@minesparis.psl.eu +33 6 47 11 52 08