Cours 6 : Perspectives

December 2, 2024

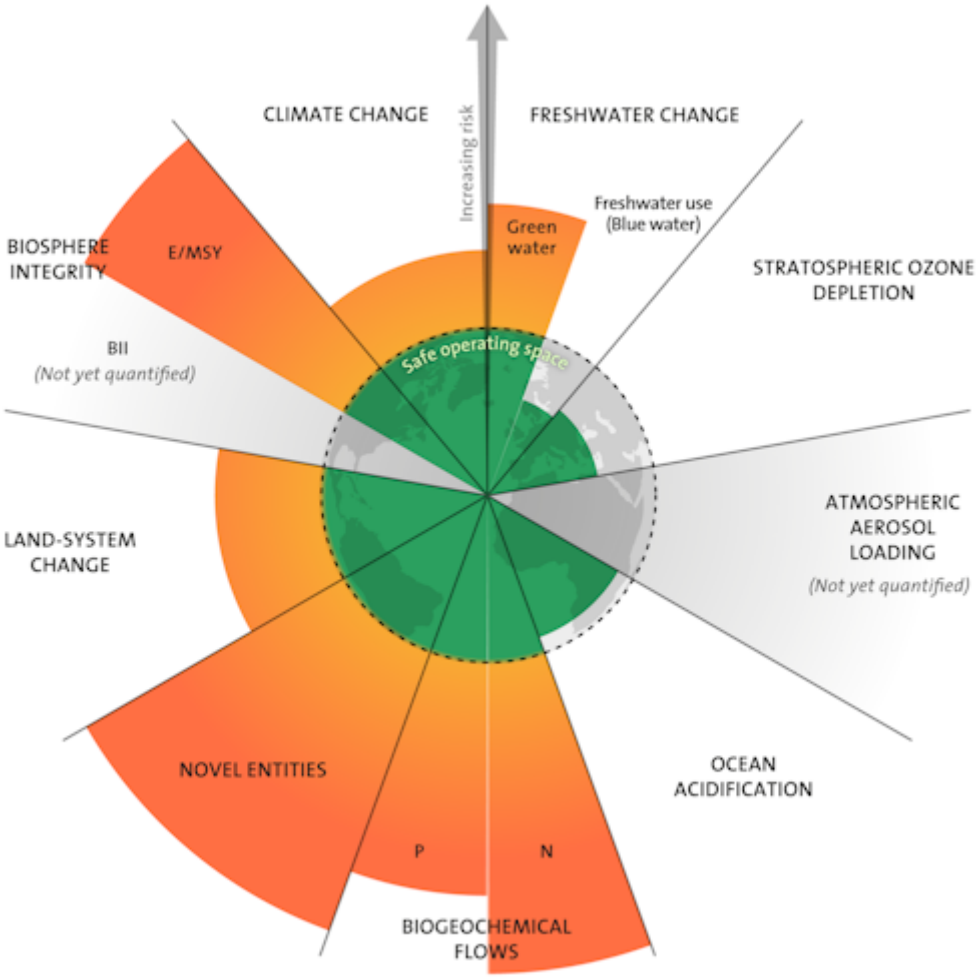
En bonus du cours d'Architecture des Ordinateurs
(année 1 : le langage machine ; année 2 : les circuits)

1 Impacts écologiques du numérique (?)

Plan :

- Constat écologique global
- Détail sur le réchauffement climatique
- Constat pour le numérique
- Problématique du matériel
  - Faire Durer
  - Sobriété (ou Frugalité)
  - (Optimiser), (Informer)
  - (Optimiser) attention aux effets rebond
  - (Informer) est-ce le métier de l'informaticien ou du communicant
- En tant que développeur
  - DD (développement durable, obsolescence, réparabilité, recyclage (?))
  - Data-Center et Serveur de Calcul vs Terminaux ? ==» organiser le logiciel en fonction (?)
  - Sobriété/Frugalité (penser à essayer sur des machines anciennes ? éviter les algos trop couteux et les machines à la pointe ? éviter les codes peu utiles)
  - Optimiser ? (Rappel)
  - propositions de solutions globales :
    * programmer pour un téléphone portable
    * avoir une démarche Eroom

2 Constats de la science (écologie)



Limites planétaires (source et illustration WKPD-2022) :

- **Changement climatique** (!!)
- Érosion de la biodiversité (!)
- Modifications des usages des sols (!!)
- Pollution chimique (nouvelles entités) (!)
- Perturbation des cycles biochimiques de l'azote et du phosphore (!!)
- Acidification des océans (OK)
- Aérosols atmosphériques (!)
- Diminution de la couche d'ozone (OK)
- **Utilisation d'eau douce** (!)

En **gras**, les domaines qui concernent le numérique.

Commentaires libres (hors transparents)

```
Rem. : le numérique est habitué à gérer les ressources avec attention
(d'où la théorie de la complexité algorithmique pour utiliser le temps
```

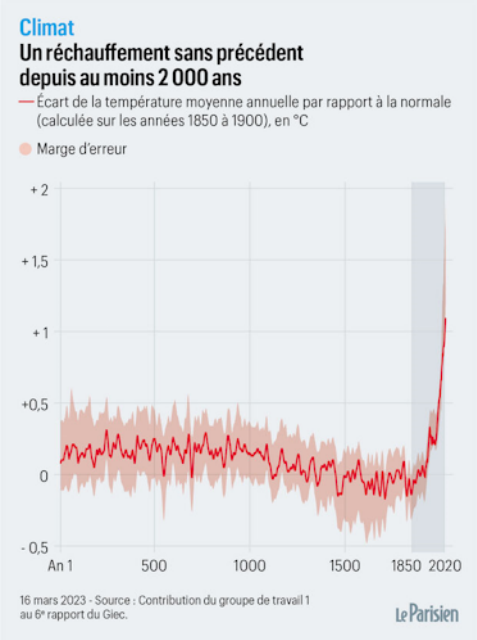
```
et l'espace à bon escient) ;
la gestion des ressources de la planète devrait donc pouvoir être prise
en compte par l'informatique

dans le désordre \[remettre un peu d'ordre ou expliquer la logique]

* parce que c'est un sujet (la gestion des ressources) familier
pour les informaticiens
* mais attention à ne pas le confondre avec une simple recherche
de la meilleure complexité algorithmique
(ce qui pourrait sembler évident et mènerait à un cours d'optimisation)
* la problématique est plus complexe (il faut avoir une approche systémique,
prenant en compte l'ensemble de la planète dans la durée)
* il y aurait même possibilité (malheureuse) de faire un contresens
en confondant optimisation
et éco-transition du numérique (voir la suite)
* tout comme en théorie de la complexité, il est trop simpliste de dire
que le meilleur programme est le plus rapide ; cela dépend des objectifs,
et parfois il faut faire des compromis entre rapidité
et mémoire pour avoir le meilleur programme en fonction d'un objectif
combinant les 2
* rem finale/additionnelle : en théorie de la complexité des circuits,
on pourrait aussi étudier le coût en terme de consommation électrique,
si ce dernier n'était pas si difficile à calculer effectivement
(la théorie est disponible, mais les calculs de coût énergétiques sont
beaucoup plus coûteux que les calculs de temps et/ou de place)
```

Conclusion de ce premier point :

- on va se concentrer sur le problème du réchauffement climatique (et un peu de l'eau)



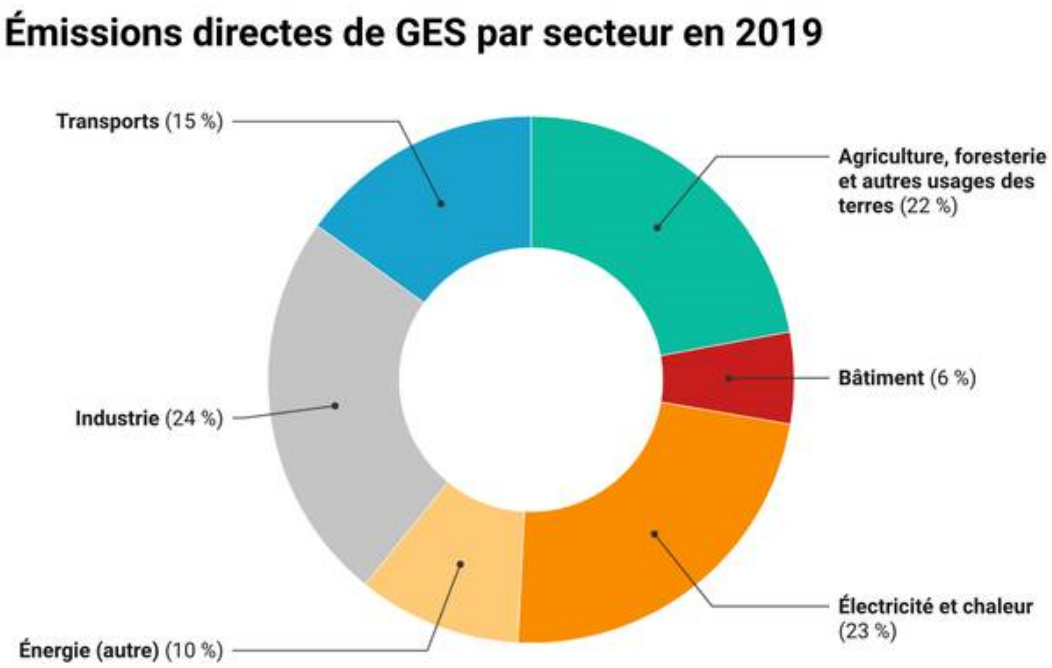
Réchauffement Climatique depuis 2000 ans (source initiale GIEC)

Commentaires libres (hors transparents)

Remarques :

* début en 1850 (1 siècle avant le premier ordinateur)
* pas accélération pour les GES (charbon avant, puis pétrole),
mais accélération de la montée des températures ces dernières années
(avec la diminution du charbon \[qui produit aussi des poussières
qui réduisent l'effet des gaz à effet de serre],
les effets des GES étaient moins visibles)
* les causes sont lointaines (la révolution industrielle),
les effets visibles/forts sont plus récents, ce n'est pas la société moderne
(post-seconde guerre mondiale) qui est (seulement) en cause,
ni l'avènement de l'informatique, ou l'essort de la santé, ou de la science,
ou de la démocratie libéro-socio-démocratico-populiste, ou de la financiarisation
de l'économie, ou sa mondialisation/globalisation
(il faut penser le changement sur 175 ans, pas juste sur ces 50 dernières années) ;
il ne faut pas tout rejeter de la société moderne parce que la planète va mal
(en particulier, l'informatique).

3 Sources des gaz à effet de serre

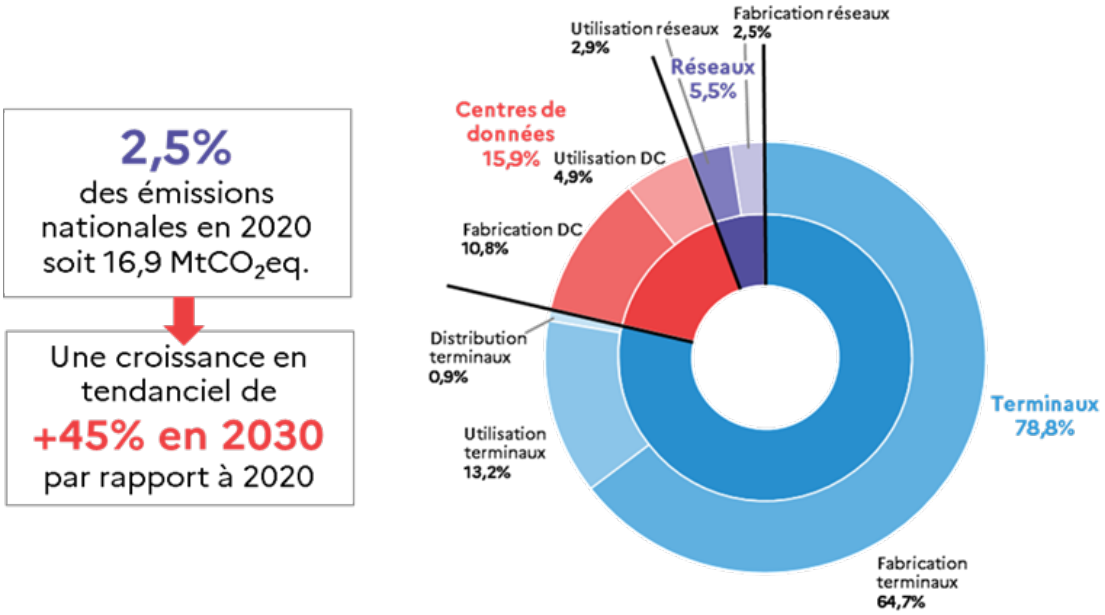


Source des gaz à effet de serre (illustration WKPD)

Le numérique se trouve dans la part concernant l'énergie (consommation à l'utilisation) et l'industrie (production du matériel) : pour 2.5 % du total.

4 Forme et place du numérique

Répartition du numérique (et des effets pour les GES) [illustration, ademe, 2020]



- 80 % : la fabrication (GES et Eau)
- 20 % : la consommation/l'utilisation (GES)

avec

- 80 % : les terminaux (ordinateurs bureau + écran, ordinateur portable (sans écran) ou avec écran supplémentaire, ou double écran, téléphones, gadgets, etc.)
- 5 % : les réseaux
- 15 % : les data-centers

Ensemble : Il faut se concentrer en premier sur le matériel (informatique)

Conclusions (pour l'informatique) :

- Il faut se concentrer en premier sur la (fabrication du) matériel (informatique), la question des économies d'énergie est (bien) moins importante.
- 6 % de croissance par an (les émissions de GES liées au numérique) : de toute façon, comme pour tout évolution exponentielle, elle ne peut pas durer longtemps, il y a toujours un moment, qui arrive rapidement, ou cela déborde (idem, quand on a un algo de complexité exponentiel, il peut marcher pour de petites données, mais ne passera quasiment jamais à l'échelle [seule exception, si le passage à l'échelle ne concerne que des petites données, ce qui est rare])

rem. :

- pour la plupart des matériels, les progrès technologiques ont amélioré (réduit) l'empreinte carbone (construction et consommation)
- sauf pour les écrans de "télévision" dont la taille (et la qualité) à explosée
- et/mais les usages ont ajouté leur coût, devant l'écran de "télévision", s'il existe encore, il n'est pas rare de voir aussi 1 ou 2 téléphones portables ou tablettes allumées qui lui font concurrence (i.e. : c'est la multiplication des écrans, chacun veut avoir le sien et faire 2-3 choses en même temps) ; finalement, le surcoût n'est pas forcément dans la construction (parfois meilleur), mais dans l'usage (qui explose)

5 Focus sur le matériel

Comment arriver à construire moins de matériel (informatique) ?

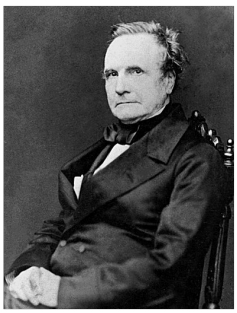
Objectifs :

- **Faire durer** (si le matériel dure 2 fois plus longtemps, il faut en construire 2 fois moins)
- **Frugalité/Sobriété** (limiter/réduire le besoin ?)
- (Optimiser), (Informer)
- autres ? (**Partager** ? => les serveurs et data-center ?)

En gras, les parties importantes (explications à suivre, développées plus loin) ; (entre parenthèse), les parties moins importantes (explications à suivre également, rapidement, juste après)

6 A propos des "optimisations"

- Attention aux effets rebonds
- Découverts/Formulés par William Stanley Jevons




sources WKPD^4

source WKPD

Commentaires libres (hors transparents)

```
Distinguer les optimisations matérielles (qui peuvent entrer en conflit direct
avec une démarche développement durable, si ces optimisations entraînent/nécessitent
des renouvellements matériels plus rapides que les renouvellements naturels) ;
à distinguer des optimisations logicielles qui n'entrent pas directement en conflit
(cela peut être indirect, cependant), mais qui posent un autre soucis très important aussi,
nommé : effet rebond.

Dans la suite de ce point, on parle surtout d'optimisation logicielle,
ce sont celles qui concernent plus directement les développeurs.

* Attention aux effets rebonds
* Découverts/Formulés par William Stanley Jevons (anglais, 19e ; (fin de) l'époque de Babbage)
* Auteur d'une calculatrice logique (le piano logique, cf. illustration source WKPD)
```

* Définition de la notion d'effet rebond (aka Paradoxe de Jevons):

```
A mesure que les améliorations technologiques augmentent l'efficacité
  avec laquelle une ressource est employée,
la consommation totale de cette ressource
  peut augmenter au lieu de diminuer (!)
```

(c'est par exemple le cas, quand la satisfaction des besoins n'a pas encore atteint les limites de la production)

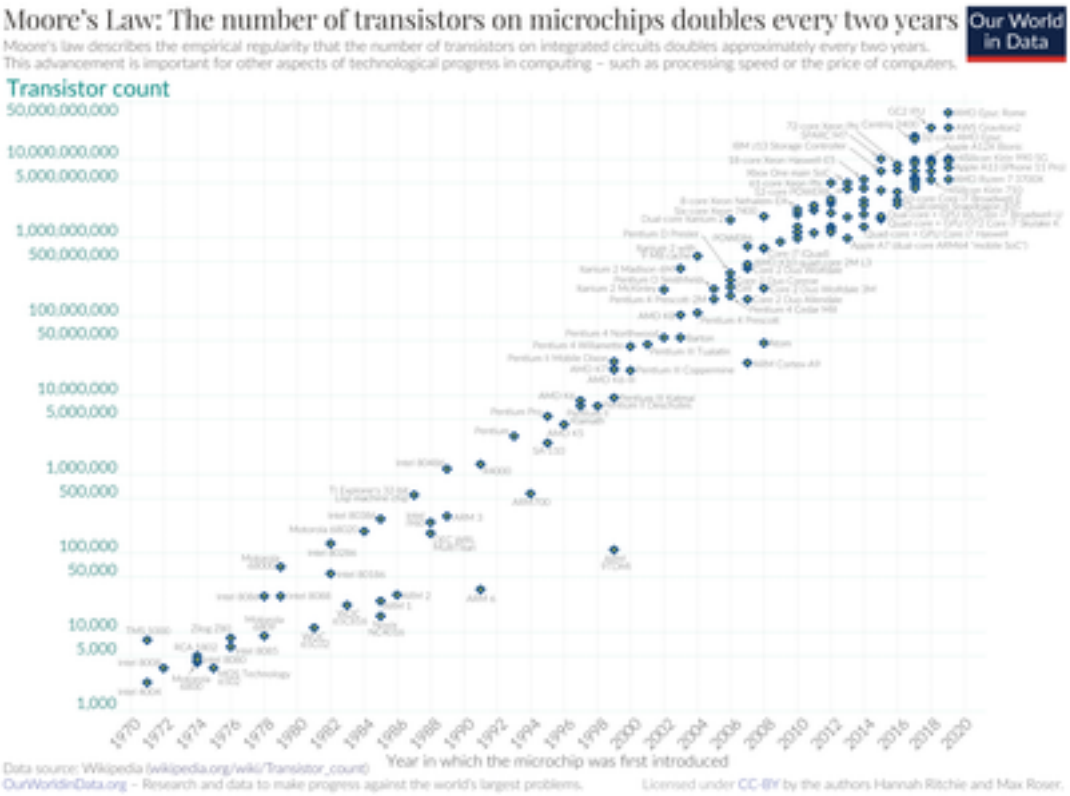
[donner peut-être des exemples ici ; des exemples arrivent ensuite, certes]

- En informatique : c'est connu sous la forme de la loi de Wirth (vs loi de Moore)

```
Les programmes ralentissent plus vite que le matériel n'accélère
```

(autre formulation : `ce que Intel te donne, Microsoft te le reprend`)

Pour la loi de Moore (source WKPD) :




```
Doublement du nombre de transistors présents
  sur une puce de microprocesseur
  tous les deux ans
```

Commentaires libres (hors transparents)

```
sur la loi de Moore : elle a constitué une source de gains (matériels) très importants,
qui auraient pu/du être à la base d'innovations et de progrès et même de comportements vertueux
(pour la consommation), mais qui (pouvaient) entrer en contradiction avec
les principes du développement durable
(construction, si nécessite renouvellement plus rapide que le renouvellement naturel)
et qui a aussi subi les contre-coût des effets rebonds)
```

Conclusion locale sur les optimisations (logicielles) :

- Optimiser, oui, si le reste a été fait (effort pour la durabilité, la sobriété)
- Dans un cadre attentif à éviter les effets rebond
- et Attention à ce que cela ne soit pas la part du colibri (il y a peut-être mieux à faire pour le colibri que d'apporter sa goutte d'eau pour éteindre l'incendie)

7 A propos du volet "Information"

- Informer : donner les indicateurs pour contrôler, étudier la gestions de ses ressources
- Expliquer : comment arriver aux solutions (les plus efficaces)
- Convertir (?) : à une démarche durable/éco-responsable (?)

- mais ce n'est pas de l'informatique (de développeur), c'est (plus) le métier des communicants

Mini-pause (fin du constat et des principes généraux)

(l'informatique n'est pas le problème (principal), ni la solution, mais doit pouvoir agir à son niveau)

Bilan/Question à mi-parcours : Que faire quand on est développeur (au courant de la part matérielle de l'informatique et de son impact sur l'environnement) ?

(Le programmeur ne va pas changer la construction des machines, et pourtant il peut agir !)

À suivre, 4 pistes (peut-être réduites) et 2 propositions de solutions (qui se veulent) plus globales.

8 En tant que développeur, pistes de solution 1 : le Développement Durable ?

Comment faire durer les machines plus longtemps ?

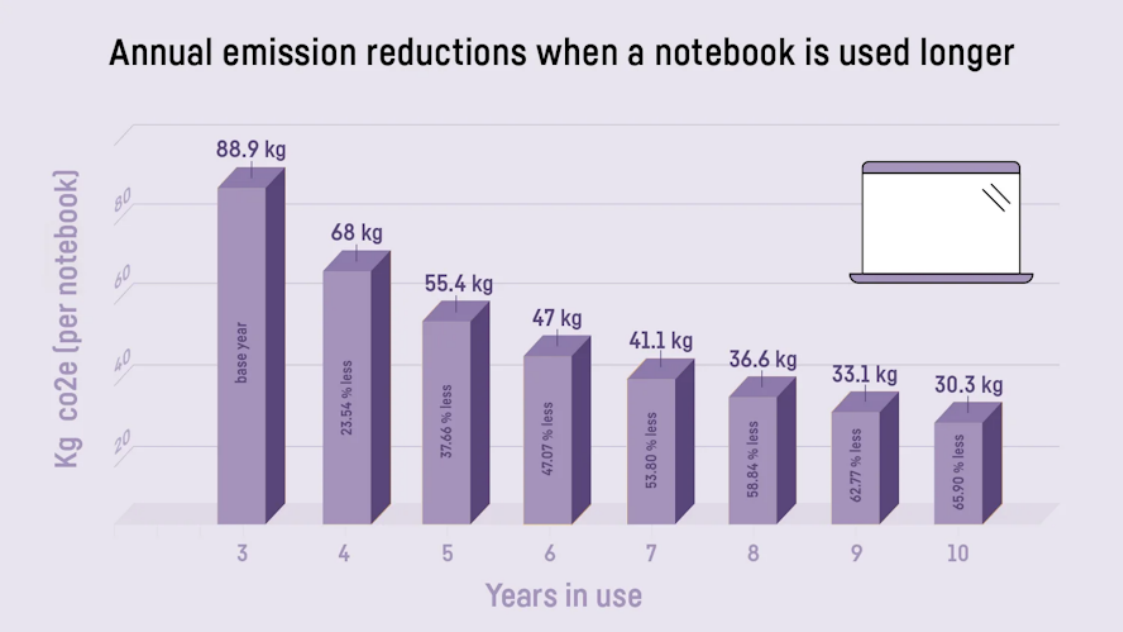
- Mot clé : Développement Durable
- Ne pas changer de matériel trop souvent (pression commerciale vs pression technique)
- problème principal (?), l'Obsolescence ((non-)programmée)
- autre piste, la Réparabilité
- autre piste, le Recyclage (?)

Commentaires libres (hors transparents)

```
pour aller au delà, raisonner de manière systémique, en cycle de vie ?
ACV : analyse du cycle de vie \[phases de conception, de production, d'utilisation,
d'élimination
; en entrées des ressources, en sortie des résidus/déchets])

* sur l'Obsolescence ((non-)programmée) : attention aux nouveautés technologiques
(pression commercial, bis), est-ce que votre machine ne tourne plus assez vite
pour les programmes courants ? est-ce que les mise à jours de sécurité
ne sont plus disponibles ? Si ce n'est pas le cas, pourquoi changer de machine ?
* autre piste, la Réparabilité : prendre soin de son matériel et le réparer
tant que c'est possible ; la réparabilité n'est pas encore l'objectif
des constructeurs mais peut le devenir, ce sera plus simple si c'est le cas
dès la conception, pour cela à l'achat, faire pression,
par ex. en prenant/payant les allongements de garantie (contre-pression des utilisateurs)
* autre piste (finale), le recyclage (?), la récupération des déchets
(déchets d'équipements électriques et électroniques \[on parle de DEEE ou D3E],
22% sur la planète en 2022, la quantité de déchets augmente plus vite que leur recyclage) :
de toute façon, si le numérique continue de croître, cela ne sera pas une solution
(au problème des ressources rares), cela retardera juste un peu le moment
où les limites seront atteintes
```

Pour bien comprendre l'effet de la durée de vie d'une machine sur l'émission de GES :



Coût annuel, d'un ordinateur (en GES, en France ; ailleurs le coût annuel de l'utilisation serait significativement plus fort (*5 ?), sauf à avoir de l'électricité verte ; mais le résultat serait sensiblement le même), Exemple :

- Durée de vie usuelle d'un ordinateur (portable sans écran supplémentaire) : entre 3 et 8 ans (il y a aujourd'hui des garanties de 7 ans)
- Pour 3 ans : 250 kg + 3 * 10 kg = 280 kg, soit ~90 kg/an
- pour 7 ans : 250 kg + 7 * 10 kg = 320 kg, soit ~45 kg/an

Commentaires libres (hors transparents)

```
Rem. :

* faire durer plus longtemps n'est pas toujours possible ou le plus efficace
* cela dépend des progrès technos matériels et/ou logiciels,
s'ils sont matériels et trop importants, trop rapides
(et c'était peut-être le cas dans les années 70-90 pour les ordis ;
dans les années 2000-20 pour les téléphones portables),
les gains apportés par les nouvelles technologies n'étaient pas au niveau
de ce que pouvaient apporter la conservation (DD) des anciens matériels,
il y avait un choix impossible entre innovation technologique et développement durable
* aujourd'hui, ordi et téléphones évoluent encore, mais à quelle vitesse (plutot lente)
et pour le matériel ? ou le logiciel ? (ralentissement de l'évolution matérielle)
* et quel sera la prochaine techno (quantique ? IA ?) pour quels apports et à quelle vitesse ?
* Ecrans et ordinateurs (ils peuvent être séparés, avoir des vies indépendantes,
attention, à ne pas les multiplier ou à l'augmentation des tailles)

Exemple de problèmes (?) :

* Arrêt prévu des mises à jours de Windows 10 prévue en Oct. 2025 (?)
```

après 10 ans \[Windows11 a pris la relève en 2021, environ depuis ~5 ans] :
https://www.greenit.fr/2024/01/16/fin-de-windows-10-37-276-687-millions-de-tonnes-eq-co2-evita

9 En tant que développeur, piste 2 : Data-Center et Serveur de Calcul vs Terminaux (peut-on partager certains coûts ? doit-on centraliser ?) ?

Constats :

- les data-centers consomment beaucoup d'électricité
- les data-centers produisent beaucoup de GES (construction)
- mais les data-centers ne représentent que 15% du numérique
  - et l'ensemble du numérique, ce n'est que 3% des émissions de CO2 (en france, 2022)
- (attention, les discours sur le sujet sont rarement neutres)

Points positifs pour les data-centers :

- Centralisme/effet de taille
- Chaleur produite ==» perdue vs exploitée ? (mais c'est 1% / industrie)
- Perspectives d'arriver à la neutralité carbone prochainement
- Sans les data-centers, il faudrait d'autres solutions (moins sobres ?)

- Développement de solutions innovantes : Edge-Computing, Gestion de l'énergie renouvelable intermittente

Points négatifs contre les data-centers :

- Effet rebond (ex. Netflix vs Dvd)
- Selon les perspectives pessimistes, les data-centers peuvent être un problème demain
- Le taux d'utilisation des data-centers est un problème
- les avantages (?) et les coûts de la redondance : 60% du coût d'un DC (pourrait être pire ?)

Conclusions :

- Améliorer les data-centers, suivre le niveau d'éco-responsabilité des data-centers, bien les dimensionner
- Trouver le bon mix traitement/donnée pour data-centers vs terminaux
- Eviter les effets rebond
- Rendez-vous en 2030 : Si neutre, alors on utilise ? mais si l'accélération continue alors on évite ?

Commentaires libres (hors transparents)

```
* les data-centers consomment beaucoup d'électricité
(et parfois d'eau pour le refroidissement \[river cooling],
mais pas toujours \[free colling]) avec un objectif d'utiliser
du renouvelable mis en avant (objectif neutralité popur 2030)

Pour les data-centers :

* Centralisme (permet des effets d'échelle, en particulier pour
la mise en place de solutions techno innovantes et éco-responsable)
et l'utilisation de constructions (récentes) efficaces/économes
(PUE proche de 1 : Power Usage Effectivemeness, mais charge ?)

Contre les data-centers :

* Effet rebond, ex. NetFlix, le streaming d'un film est vertueux comparé
à l'achat d'un DVD (pour un usage unique), mais le nb de films/séries
vus a explosé !
* Le taux d'utilisation des data-centers est un problème
(il est faible ==>> mauvais pour la planète ;
il est fort ==>> le service rendu est de moins bonne qualité,
la concurrence peut gagner des parts de marché avec un taux plus faible
 \[comme pour la tragédie des communs, l'auto-organisation du secteur peut
 mener à des solutions non profitable à la planète])

Conclusions :

* Améliorer les data-centers, suivre le niveau d'éco-responsabilité
des data-centers (pue, charge, neutralité carbone des sources d'énergie,
localisation et autres), en particulier bien les dimensionner
(70% des entreprises ne connaissent pas la charge de leurs machines,
```

et seulement 12% des machines sont avec une charge > 60%)

10 En tant que développeur, piste 3 : la Sobriété ?

- penser à essayer sur des machines anciennes
- éviter les algos trop coûteux ou les modes irresponsables (boucles/défilements infinis)
- éviter les machines ou les technos à la pointe
- éviter les modes irresponsables (ex. : les boucles infinies comme le défilement à l'infini)
- attention aux librairies obèses et aux codes peu utiles (obésiciel : logiciel-obèse)
- prévoir les mises à jour de sécurités indépendantes des ajouts de fonctionnalités (plus coûteuses)
- prévoir des ajouts de fonctionnalités (rares) indépendantes

mais, attention :

- un peu de mesure
- parfois, il faut aussi des algos coûteux, des machines performantes, etc.
- il faut réfléchir globalement (pensée systémique [ex. étudier pour apprendre ou pour le diplôme ?])

Quelques contre-exemples de frugalité (exemple à ne pas suivre, rendu possible par effet rebond de l'augmentation des performances des machines, sans conservation/réduction de leur usage courant) :

- VsCode et les sauvegardes ou la compilation en permanence
- YouTube vs ChatGpt vs StackOverflow vs Wikipédia
- Algorithmes de suggestions, addictifs, à l'infini

Commentaires libres (hors transparents)

```
détail pb, vs-code (pour afficher les erreurs au fur et à mesure)
à distance sur im2ag-turing (limites atteintes !)
solution, compiler en local ? limiter la compilation permanente (en directe),
idem pour les sauvegardes ? à n'utiliser que s'il y a une plus value (pour l'apprentissage)
```

11 (En tant que développeur, piste 4 : Optimiser ? (Rappel))

(rappel) :

- Attention aux effets rebond
- Optimiser seulement si DD et Sobre déjà fait (c'est plus important)
- Rappel de L2 : pour Optimiser, le **developpeur** peut aider, le **compilateur** et la **machine** aussi ; le plus efficace, c'est quand tout le monde s'y met et respecte les contraintes des autres
  - Eviter les ruptures de séquences (conditionnelles) pour éviter de rompre les pipe-lines matériels
  - Prévoir et faciliter les apports des caches (matériels, réseaux, logiciels, etc.) et de la hiérarchie mémoire (favoriser les principes de localité spatiale et temporelle)

- Utiliser les potentialités des multi-coeurs (plutôt que d'avoir un ordi surdimensionné // ou avoir un ordi à la puissance bien dimensionnée !)
- Reduire les contraintes de précédence de l'exécution dans le desordre pour améliorer les optimisations du compilateur
- Utiliser la récursivité (et les fonctions) à bon escient, pour que cela soit un gain et non pas un surcoût
- Ajuster la taille des données (numériques ou textuelles) à leur réelle nécessité

Pour votre information :

Total

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Efficacité des langages de programmation

Si vous voulez un exemple :
* voir ici

12 Propositions de solution globale 1 : Développer pour les téléphones portables (?)

- programmer pour un téléphone portable (un peu ancien) ou des terminaux/ordinateurs (un peu ancien) peu puissants (avec écran limité),
- les ressources peuvent être déportées (sur les serveurs et data-centers ?)
- source : Numérique Eco-Responsable, https://www.greenit.fr/
- 115 bonnes pratiques pour le developpement web : https://collectif.greenit.fr/ecoconception-web/115-bonnes-pratiques-eco-conception_web.html

13 Propositions de solution globale 2 : Démarche erooM (?)

Avoir une démarche erooM :

```
erooM ,
  c'est la décision d'optimiser le logiciel existant
  d'un facteur deux
  tous les deux ans
```

Dans la pratique, organiser son travail en 3 temps (au cours de la vie du logiciel) :

- temps 1 : Développer (sobre et durable)
- temps 2 : Stabiliser
- temps 3 (récurrent) : Optimiser (pour revenir à une dépense énergétique soutenable/compatible avec les limites planétaires à fonctionnalités équivalentes), objectif : diminuer par 2 tous les 2 ans le coût GES.

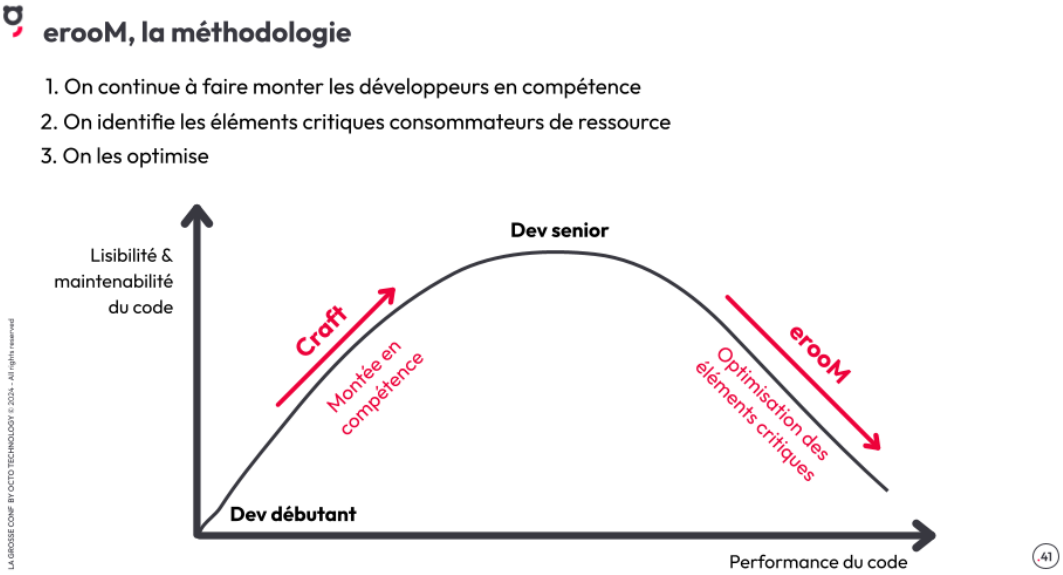
ou

- travailler avec une idée des ressources limitées
- ressources à ne pas dépasser, compatibles avec les vieilles machines
- et améliorer encore (pour revenir à une dépense énergétique soutenable)

Remarques :

- une différence importante entre loi de Moore et démarche erooM : matériel vs logiciel.
- mais des similarités (à surveiller) : sensible aux effets rebonds

Illustration



source blog Octo https://blog.octo.com

Commentaires libres (hors transparents)

```
erooM ? (le nom n'est pas important, c'est Moore à l'envers ;
l'important c'est d'arriver à faire décroître l'empreinte carbone,
ce qu'aurait du avoir comme conséquence la loi de Moore s'il n'y avait pas
eu d'effets rebond
ou si elle n'avait pas été matérielle et (trop) rapide)

La loi de Moore a porté l'innovation et le numérique depuis 50 ans (et +),
mais elle s'essoufle (et entre en collision avec les limites planétaires) ;
pour ne pas perdre ses bienfaits (et la planète en aura peut-être besoin,
c'est le Green-By-IT),
il faut remplacer ces gains technologiques/matériels par des gains logiciels
(renverser la loi de Wirth, si c'est possible)
```

14 Conclusion et Remarques finales

- L'important, c'est (vraiment) du côté machine (des utilisateurs) !
  - il faut les faire durer, c'est l'essentiel, aujourd'hui (c'est le côté construction, 80% !)
  - (et en particulier éviter toutes les formes d'obsolescence induites par nos comportements d'informaticiens, développeurs, entre autre;
    - y réfléchir à travers le cycle de vie des logiciels ;
    - comprendre tous les liens qui unissent les logiciels aux machines ;
    - les ressources nécessaires au logiciels, les mises à jours sur des machines parfois anciennes ou moins puissantes, les obèsiciels, etc.)
  - en second, on peut aussi viser une consommation électrique qui reste/devienne sobre

  - surtout ne pas chercher à croître sans limite (pensez aux ressources matérielles et énergétiques ou temporelles comme des limites à respecter, sur le long terme)
    * (on doit pouvoir se satisfaire de la croissance de Moore ou de erooM, et éviter les 6% de croissance actuels)
  - pour cela, le travail le plus important (pour l'éco-transition du numérique) n'est pas (seulement) au moment du développement
    * mais peut-être (/surtout) en amont et en aval, dans les choix (d'architectures matérielles et logicielles) et les infrastructures utilisées (le mix Data-Center - Terminaux) qui décideront des développements
  - (et attention de ne pas engendrer des effets rebonds pour les optimisations)
  - (le reste, les usages, nous concernent, mais, en tant que développeur, nous n'en sommes pas les principaux/seuls responsables)
- Cette éducation à la transition écologique du numérique peut passer -entre autre- par les cours où l'on parle des machines (et des couches basses de l'informatique),
  - mais aussi par les cours de dvpt (mais attention, c'est à la fois plus clair[1] mais moins direct [2])
  - pour les cours d'archi, c'est une bonne raison pour protéger et promouvoir ces cours
    * car eux, en particulier, permettent de ne pas invisibiliser les machines en informatique
- En outre : si en informatique, l'effort sera réduit, ce ne sera peut-être pas le cas pour les autres secteurs (transport/agriculture/logement), la transition climatique c'est un vrai enjeu (mais pour la couche ozone, la planète a réussi à s'en sortir, donc tout espoir est permis)
- Remarque finale : Programmer de manière éco-responsable (**vert**ueusement) peut/doit être bénéfique à la planète, à l'utilisateur, à l'entreprise (rechercher un équilibre de Pareto ?) : la planète ira mieux, pour l'entreprise cela pourra participer à son image de marque (attention, pas du green-washing) et l'utilisateur se portera mieux dans sa tête.

15 Références

- Pour une transition numérique écologique, Rapports d'information au Senat n° 555, Juin 2020, Guillaume CHEVROLLIER et Jean-Michel HOULLEGATTE
- Green IT : les clés pour des projets informatiques plus responsables, 2022, Margerie Guilliot, Raphaël Lemaire, Sylvain Revereault, ENI édition
- Ecoconception web : les 115 bonnes pratiques, 4ème édition, mai 2022, Frédéric Bordage / GreenIT.fr, Ed. Eyrolles
- Datacenter, Maîtriser et optimiser son impact environnemental, Livre Blanc, Alliance Green IT
- Démarche erooM : https://blog.octo.com/octo-article-de-blog-4

[1] c'est peut-être plus clair/direct de parler des problématiques de mises à jour dans un cours de devpt logiciel, mais [↑2]

[2] c'est moins clair de parler des problématique écologiques dans un cours de devpt logiciel, car dans ces cours les machines sont hors du périmètre usuel et le prb écologique passe essentiellement par la question des machines en info, donc le pb écologique semble déconnecté de ce cours (la force du lien entre les différentes formes de dvpt et le pb écologique risque d'être faible).