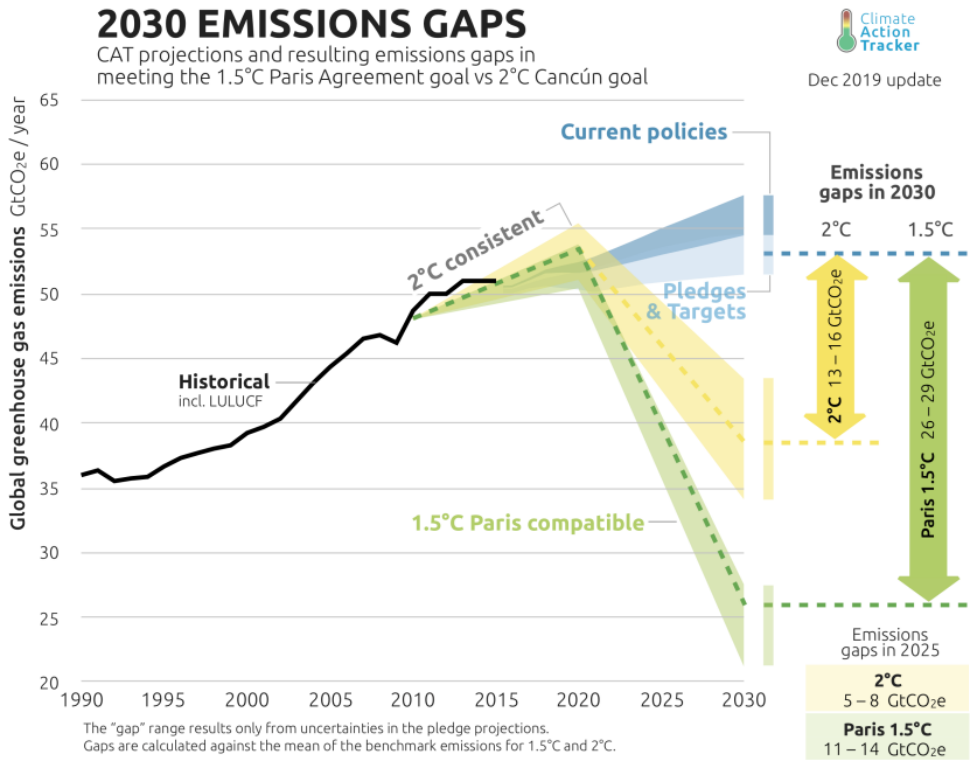
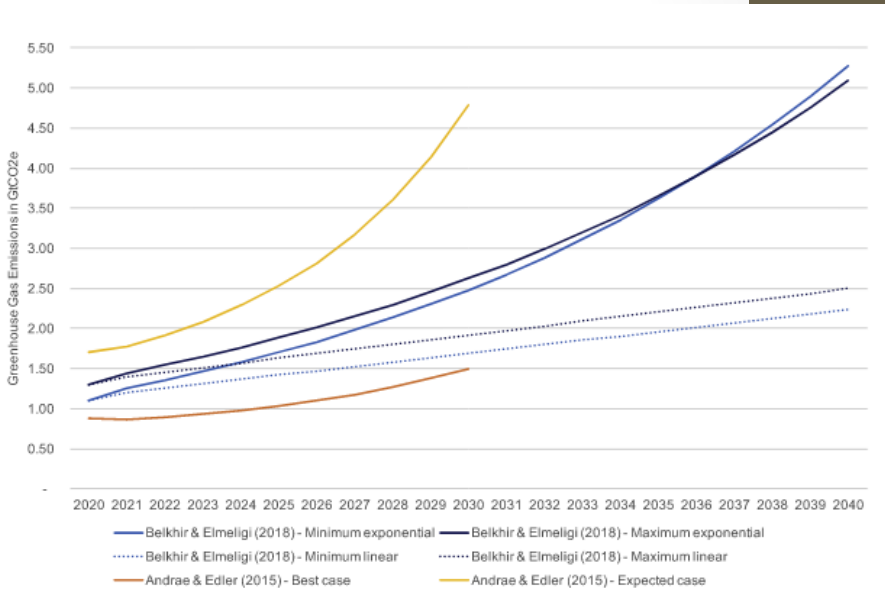
Programmation Orientée Objet en C++

Chapitre 10 : Écoconception Logicielle

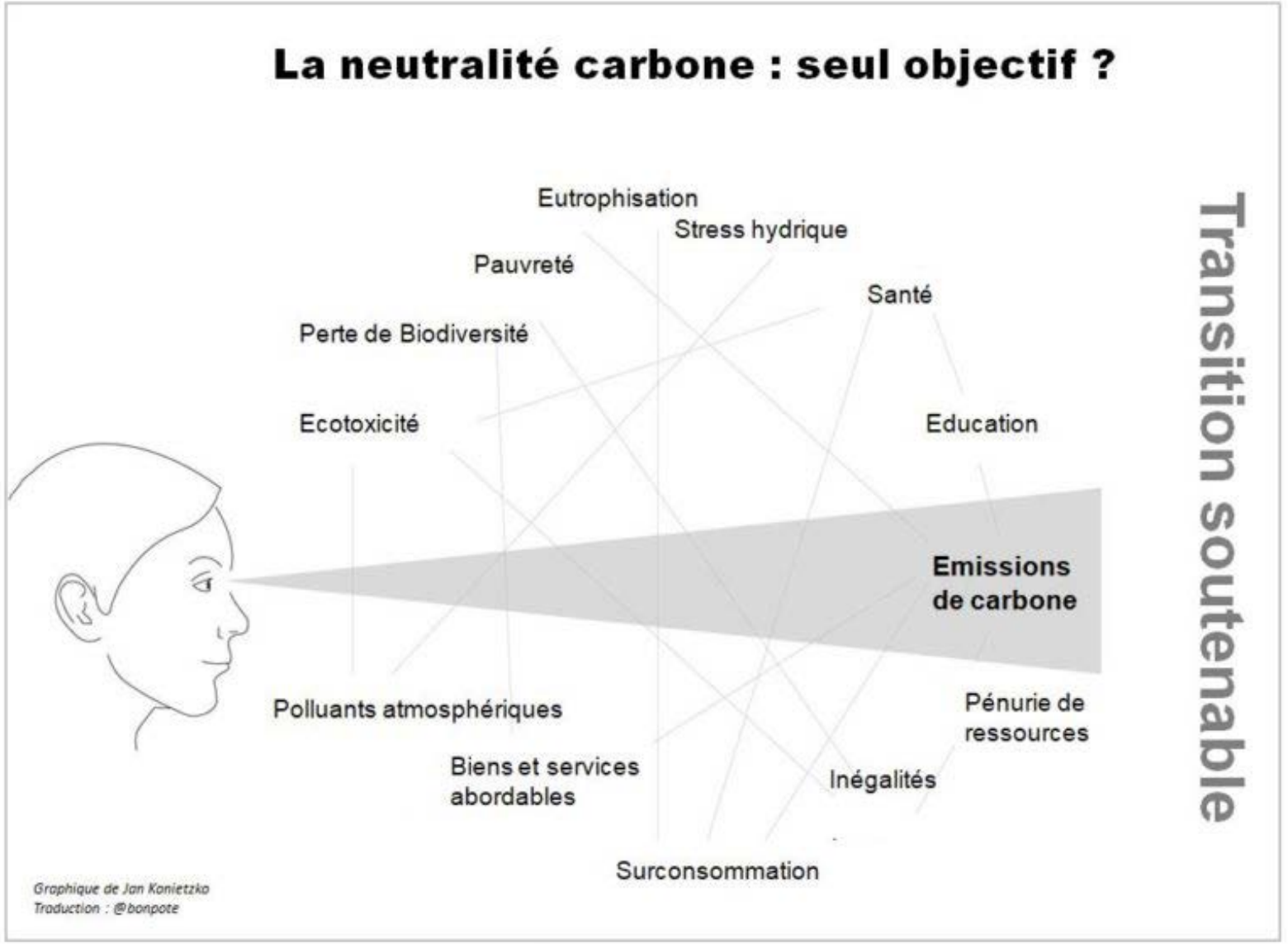
Emmanuelle Frenoux
Emmanuelle.Frenoux@universite-paris-saclay.fr

Rappel : émissions de GES du numérique





Rappel : effet œillères



Rappel toujours : ACV smartphone

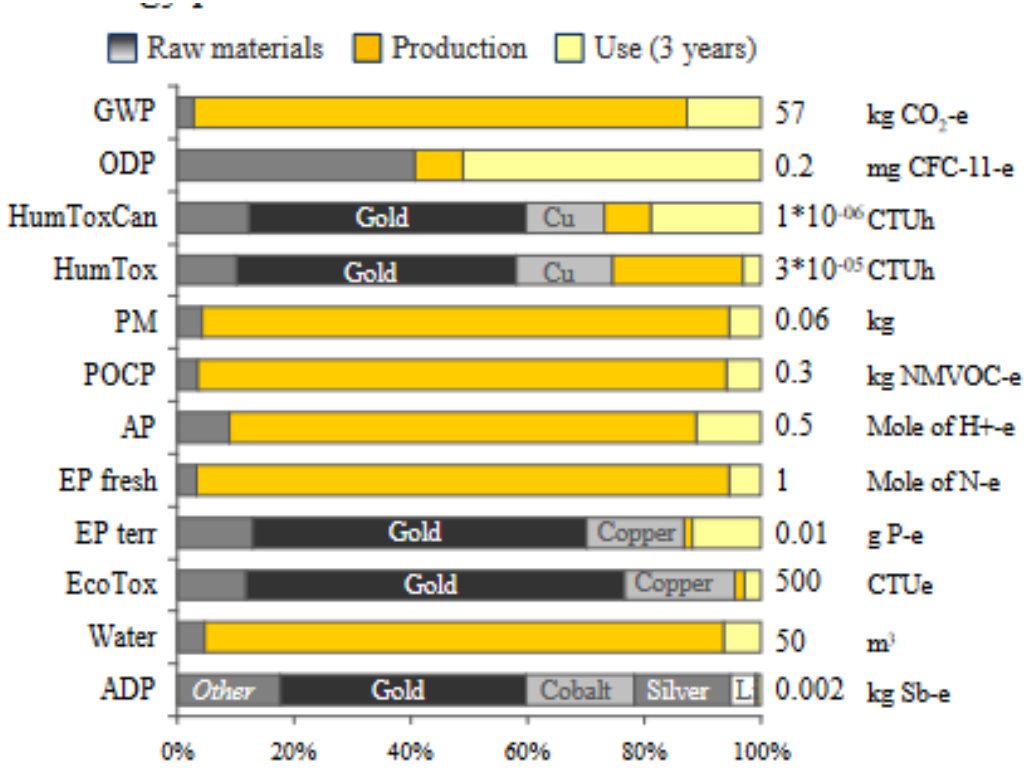
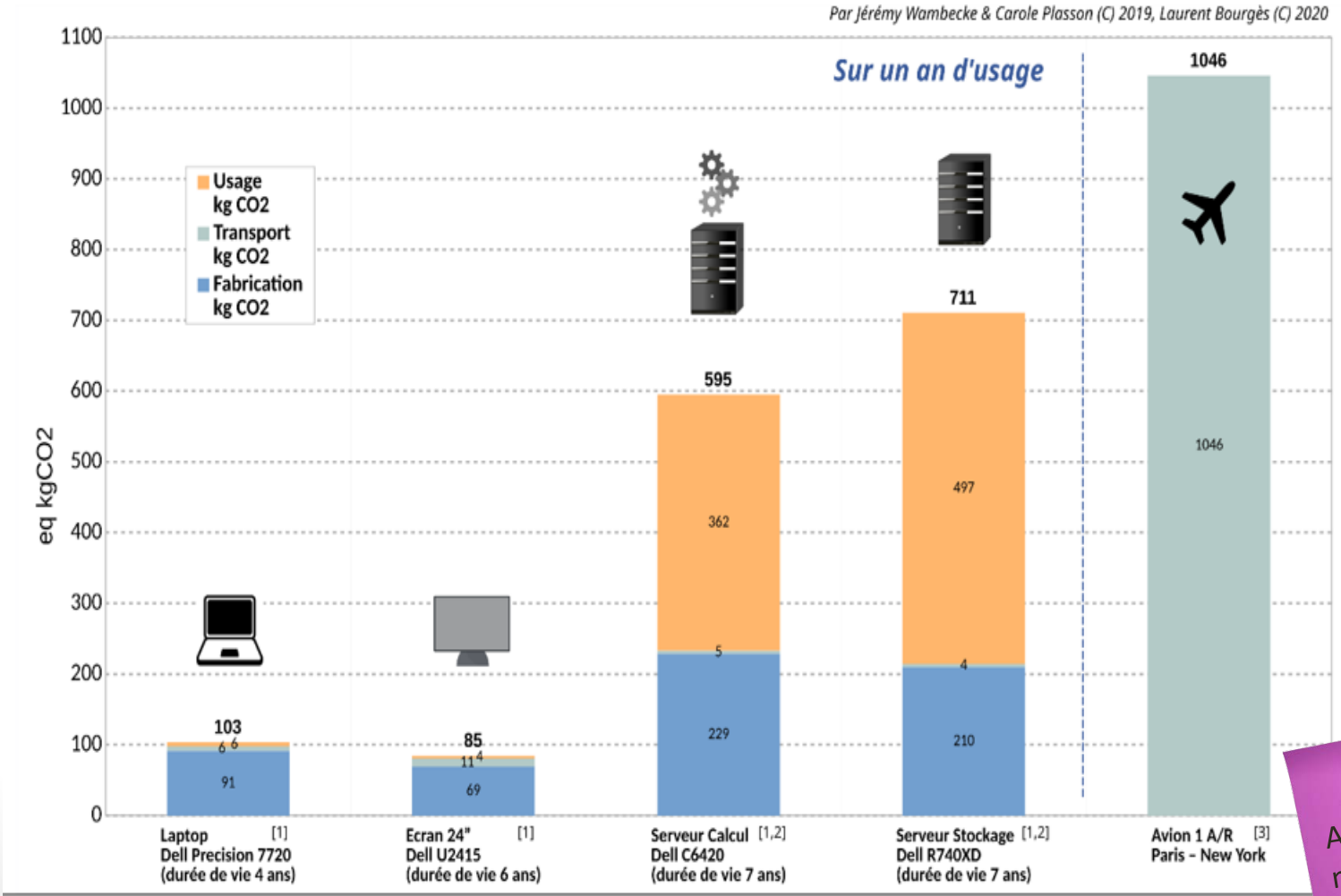


Fig. 4 Total life cycle result for all impact categories for smartphone Z5 with accessories using Ecoinvent database and adopting a 50/50 recycling approach with 19% recycling of gold assumed.

Life Cycle Assessment of a Smartphone , Ercan et al, Conf.ICT for Sustainability 2016.
https://www.researchgate.net/publication/308986891_Life_Cycle_Assessment_of_a_Smartphone

Rappel encore et toujours : matériels



[1] Données Fiches Dell (usage corrigé pour usage FR) :
(https://www.dell.com/learn/us/en/uscorp1/corp-comm/environment_carbon_footprint_products)
[2] Usage à partir de la consommation moyenne (Berthoud et al. 2020) d'un noeud = 275W (C6420), 375W (R740XD) (https://hal.archives-ouvertes.fr/hal-02549565)

[3] https://eco-calculateur.dta.aviation-civile.gouv.fr/
Facteur d'impact : 0,108 kgCO2e/kWh (FR)

Attention au mix énergétique

Les TIC, c'est aussi du code

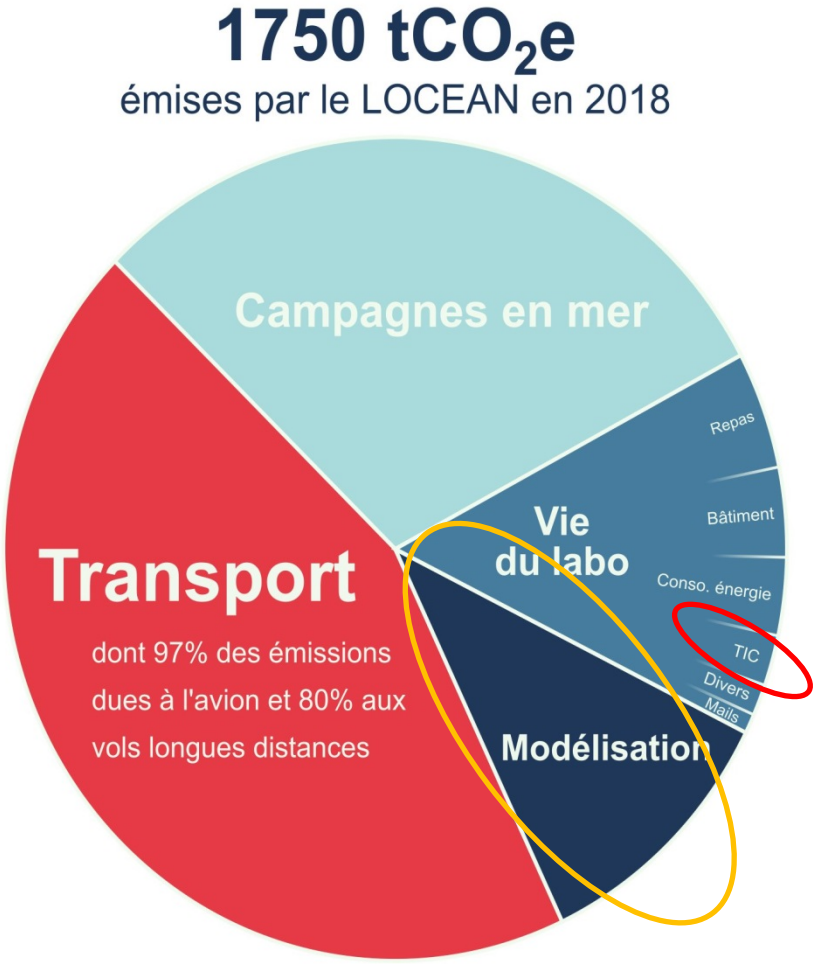
Des millions d'applications, de logiciels de formats différents : distribué/web/cloud, virtualisé, sur le store (Apple, Play), réseau (Software Defined Network & Network Function Virtualization)

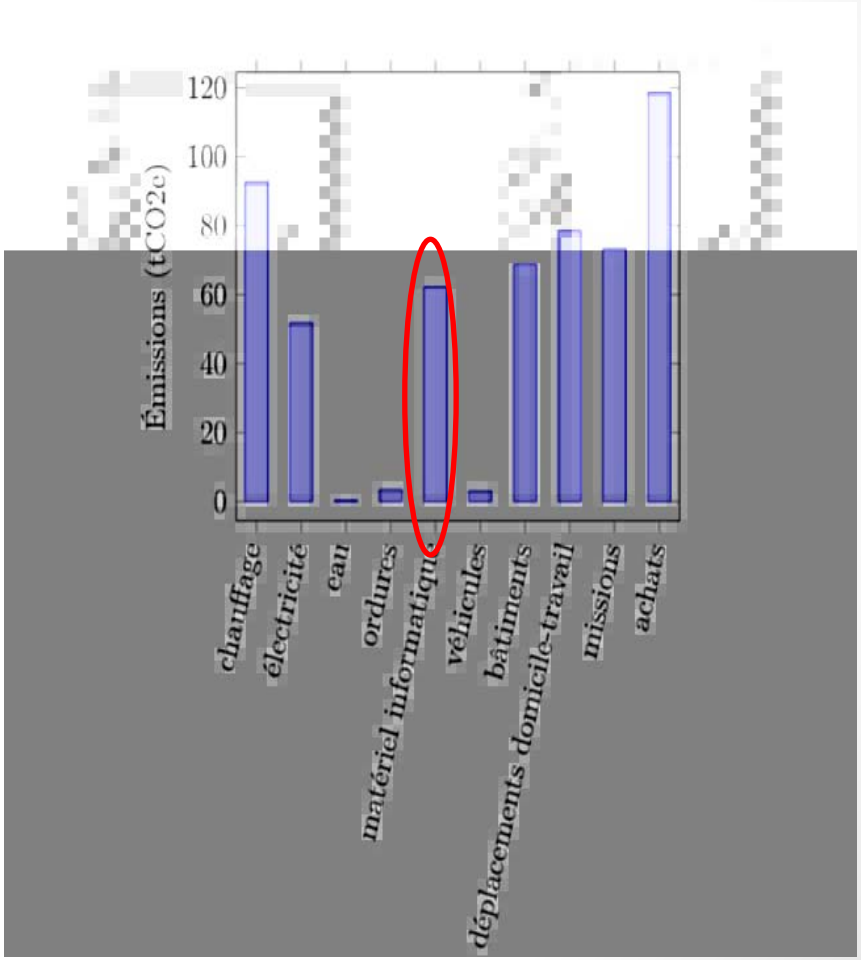
VOLUMES REPRÉSENTÉS :

- Plus d'un million de logiciels par store
- Environ 30 applis par smartphone
  - Une application sur 4 non utilisée
  - 59 % des applications utilisées une seule fois

- Application simple : une centaine de lignes de code
- Androïd : 10 millions, Noyau Linux : 15 millions
- Windows 7 : 40 millions, Office 2013 : 45 millions
- Services Web Google : 2 milliards

Prise en compte à l'échelle d'un laboratoire





LOCEAN : UMR, 187 personnes, INSU.

LIMSI : UPR, 169 personnes, INS2I.

À l'échelle d'un labo d'astronomie

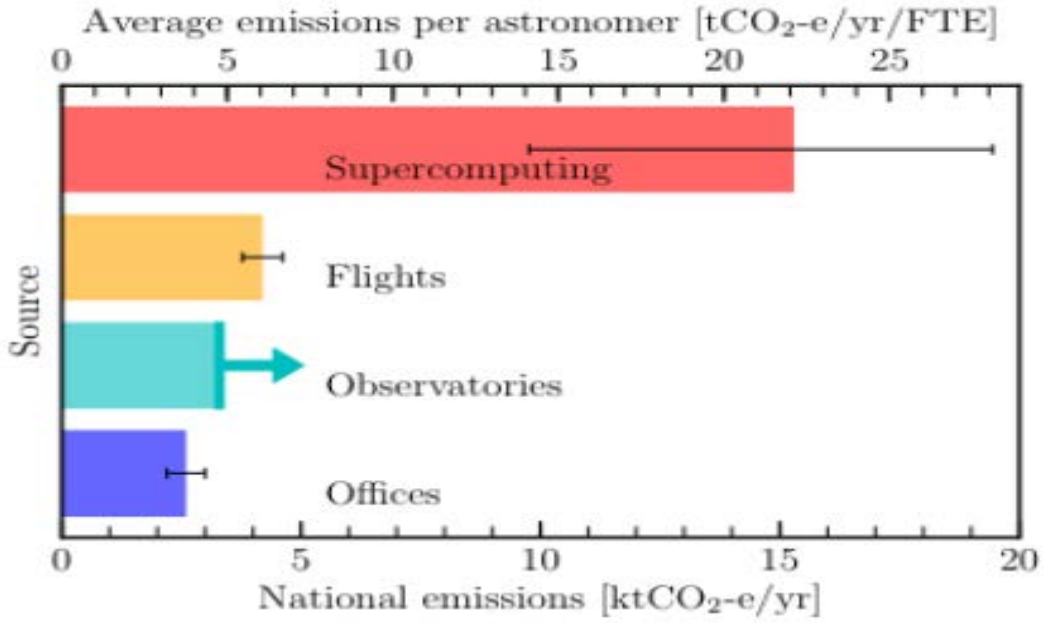
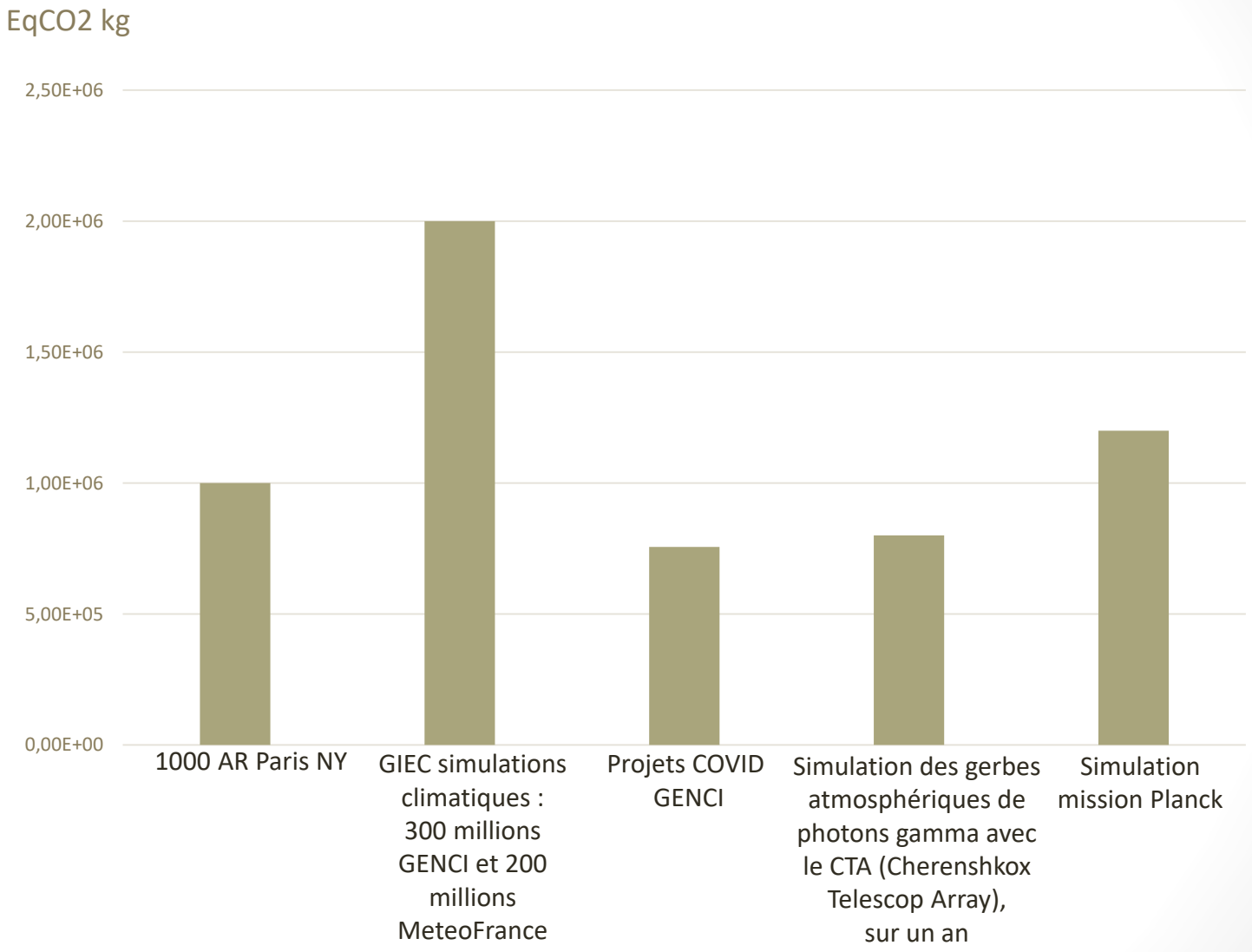


Figure 2: Breakdown of the four sources of Australian astronomers' emissions considered in this work. Error bars provide an estimate of our uncertainties, but should not be interpreted as formal confidence intervals. The value for observatories is a lower limit. 'Per astronomer' refers to the 691.7 FTE including PhD students, postdocs, and senior researchers.

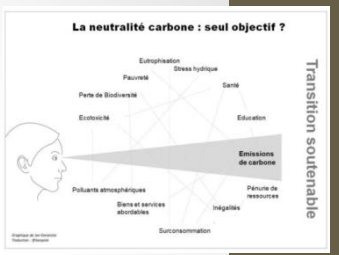
Le Calcul, à l'échelle d'un pays



Estimation faite en utilisant le facteur de conversion 1heure de calcul = 0.004 kg EqCO2
Graphe : AC Orgerie, EcoInfo
Sources : chiffres GENCI et estimations d'Ingénieurs ayant travaillé sur les différents projets.

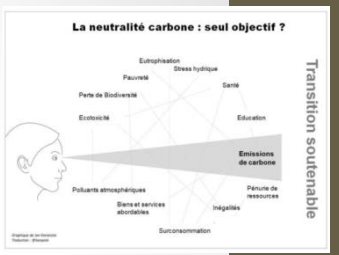
Comment fait-on pour mesurer ça ?

Outils physiques de mesure de consommation



- wattmètre, ampèremètre
- PDU (Protocol Data Unit ou Unité de données de protocole)
  - unité de mesure des informations échangées dans un réseau informatique
  - permet de monitorer des machines
  - est une « mini-machine » physique
- On peut tester le monitoring en lançant des tâches sur GRID5000

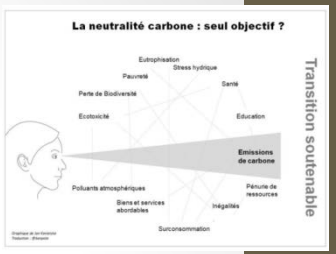
Outils logiciels de mesure de consommation



- Power API
- Intel Power Gadget
- Mac Power Meter (comparaison logiciel/wattmètre)
- Dans des conditions « bornées », on sait également calculer le transit des données…
  Un exemple de mesure du coût du transit des données sur Renater :
  https://ecoinfo.cnrs.fr/wp-content/uploads/2020/12/Rapport-revise-1Go-VF02-2021.pdf
  - PB : dans la vraie vie, le système n'est pas borné du tout !

https://software.intel.com/content/www/us/en/develop/articles/intel-power-gadget.html.
https://gitlab.inria.fr/guenneba/mac-power-meter/-/tree/master.
https://pypi.org/project/powerapi/.

Outils de mesure en ligne



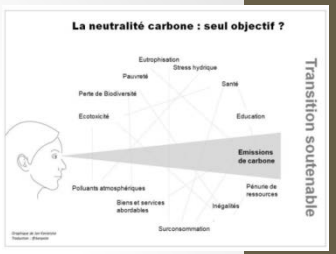
Mesure d'impact d'une page web…

- Carbonalyser : extension du navigateur calculant le carbone émis pendant un temps d'utilisation du navigateur
- Ecometer : analyse l'impact environnemental d'un site
- Ecoindex : analyse l'impact environnemental d'un site

Attention, certains outils sont critiqués pour leur imprécision.

https://github.com/carbonalyser/Carbonalyser
http://www.ecoindex.fr/
http://www.ecometer.org/

Évaluer son parc, son bilan GES



- **Évaluer son matériel :** Ecodiag
  https://ecoinfo.cnrs.fr/ecodiag-calcul/
- **Effectuer le bilan GES du labo :** GES1point5
  https://www.labos1point5.org/ges-1point5

➢ On peut aussi [faire] auditer son centre de calcul/de données

Focus Centres de calcul/de données

- PUE : Power Usage Effectiveness
  - mesure de l'efficacité énergétique (rendement)
  - Développé par le consortium The Green Grid
  - $PUE = \frac{Energie\ Totale\ Consommée\ par\ le\ Centre}{Energie\ Totale\ Consommée\ par\ les\ équipements\ Info}$
- Attention : on ne regarde plus que l'usage !
- Peut-on retirer la chaleur fatale dissipée "intelligemment" du PUE ?
- Et pour l'énergie issue de panneaux solaires ?
- Valeur idéale : 1
- Estimation du coût d'une heure.cœur de calcul : https://hal.archives-ouvertes.fr/hal-02549565

Réf : https://www.thegreengrid.org/

Bien urbaniser un Centre (calcul/données)

- Une bonne urbanisation diminue la consommation d'énergie
- Optimiser la température de fonctionnement
- Respecter les couloirs chaud et froid
- Améliorer la circulation d'air
- Monitorer raisonnablement
- Mesurer sur un tableau électrique dédié
- Mesurer son PUE
- Bien dimensionner (puissance, nombre de machines, etc)

Choisir son supercalculateur : toi aussi, t'es dans le Top 500 ?

TOP500 : projet d'évaluation des 500 plus gros supercalculateurs en termes de performances sur des benchmarks d'analyse numérique.

Depuis 2007 : Green 500 évalue l'efficacité énergétique.

https://top500.org/

Quelques bonnes pratiques en éco-conception de services numériques

Je code : les bonnes pratiques en écoconception de service numérique à destination des développeurs de logiciels

Source : https://hal.archives-ouvertes.fr/hal-03009741v4/document

C’est quoi, un service numérique ?

- C’est :
  - information (données)
  - traitements (algorithmes, filtrage, simulation)
  - échanges d’informations
  - interfaces utilisateurs
- Ça repose sur :
  - infrastructures logicielles (applis, outils, bibliothèques, protocoles)
  - infrastructures matérielles (serveurs, équipements réseau, terminaux, capteurs)
  - personnes (développeur·euses, administrateur·ices systèmes et réseaux, chef·fes de projet, chercheur·euses)

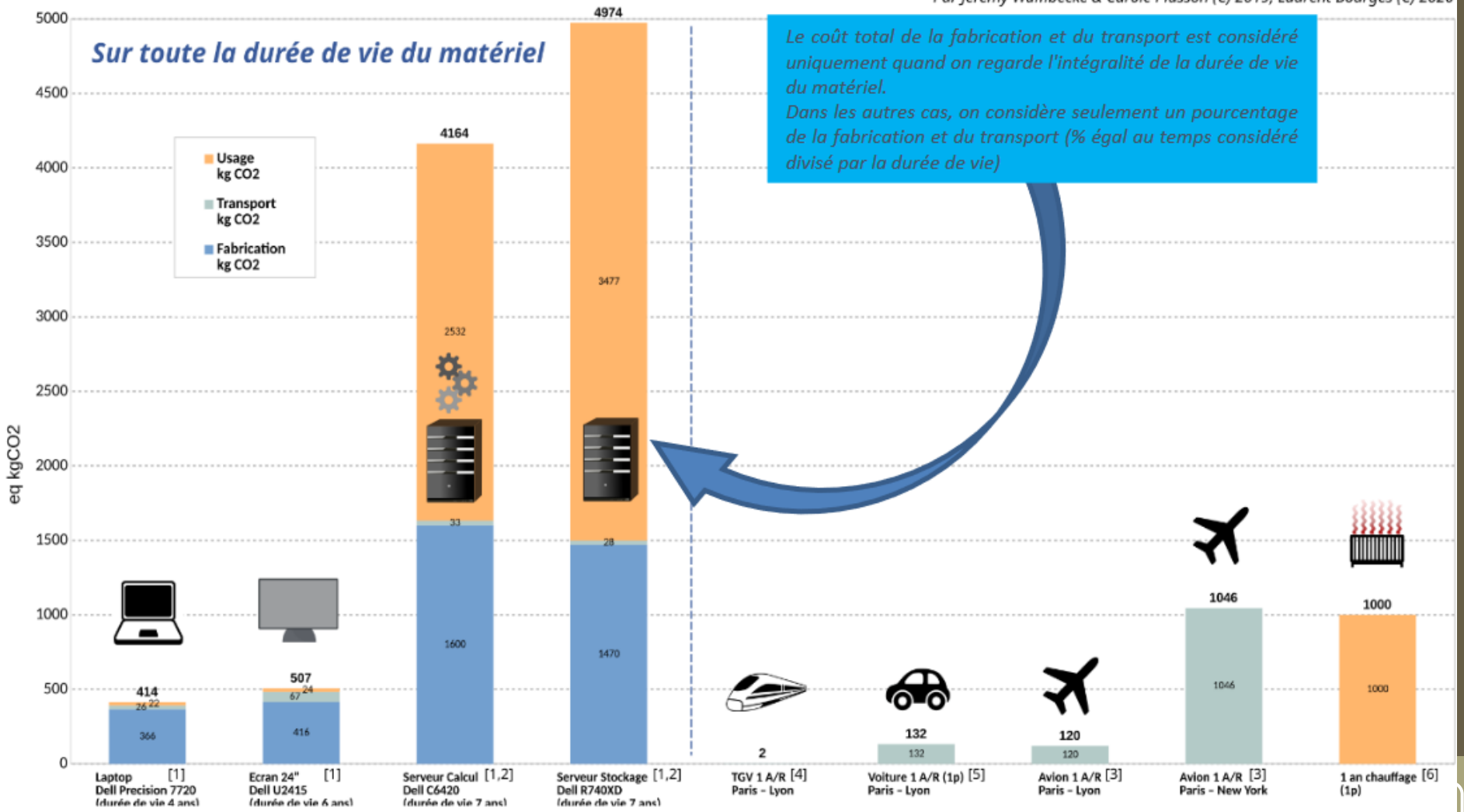
Exemple de service numérique : workflow d'une simulation numérique

- Briques
  - préparation des données d'entrée
  - transfert des données d'entrée vers la plateforme de calcul
  - calcul sur la plateforme
  - transfert des données vers la plateforme de post-traitement
  - post-traitement, stockage et dissémination des données de sortie du calcul
  - analyse et l'exploitation des données
- Leviers de réduction des impacts
  - Limiter les transferts de données
  - Choix plateformes de calcul, de pré et post-traitement
  - Post-traiter au plus proche du lieu de création
  - Limiter les données d'entrée et/ou de sortie
  - Choisir des briques logicielles externes ou à redévelopper en interne

Pourquoi faire de l'éco-conception logicielle ?

Comparatif d'émissions CO2

Par Jérémy Wambecke & Carole Plasson (C) 2019, Laurent Bourgès (C) 2020



[1] Données Fiches Dell (usage corrigé pour usage FR) : (https://www.dell.com/learn/us/en/uscorp1/corp-comm/environment_carbon_footprint_products)
[2] Usage à partir de la consommation moyenne (Berthoud et al. 2020) d'un nœud = 275W (C6420), 375W (R740XD) (https://hal.archives-ouvertes.fr/hal-02549565)
[3] https://eco-calculateur.dta.aviation-civile.gouv.fr/

[4] https://ressources.data.sncf.com/explore/dataset/emission-c02-tgv/table/
[5] Trajet de 473km, pour une voiture émettant 0,140 kg CO_2/km
[6] https://www.insee.fr/fr/statistiques/fichier/1281320/ip1445.pdf

Facteur d'impact : 0,108 kgCO_2/kWh (FR)

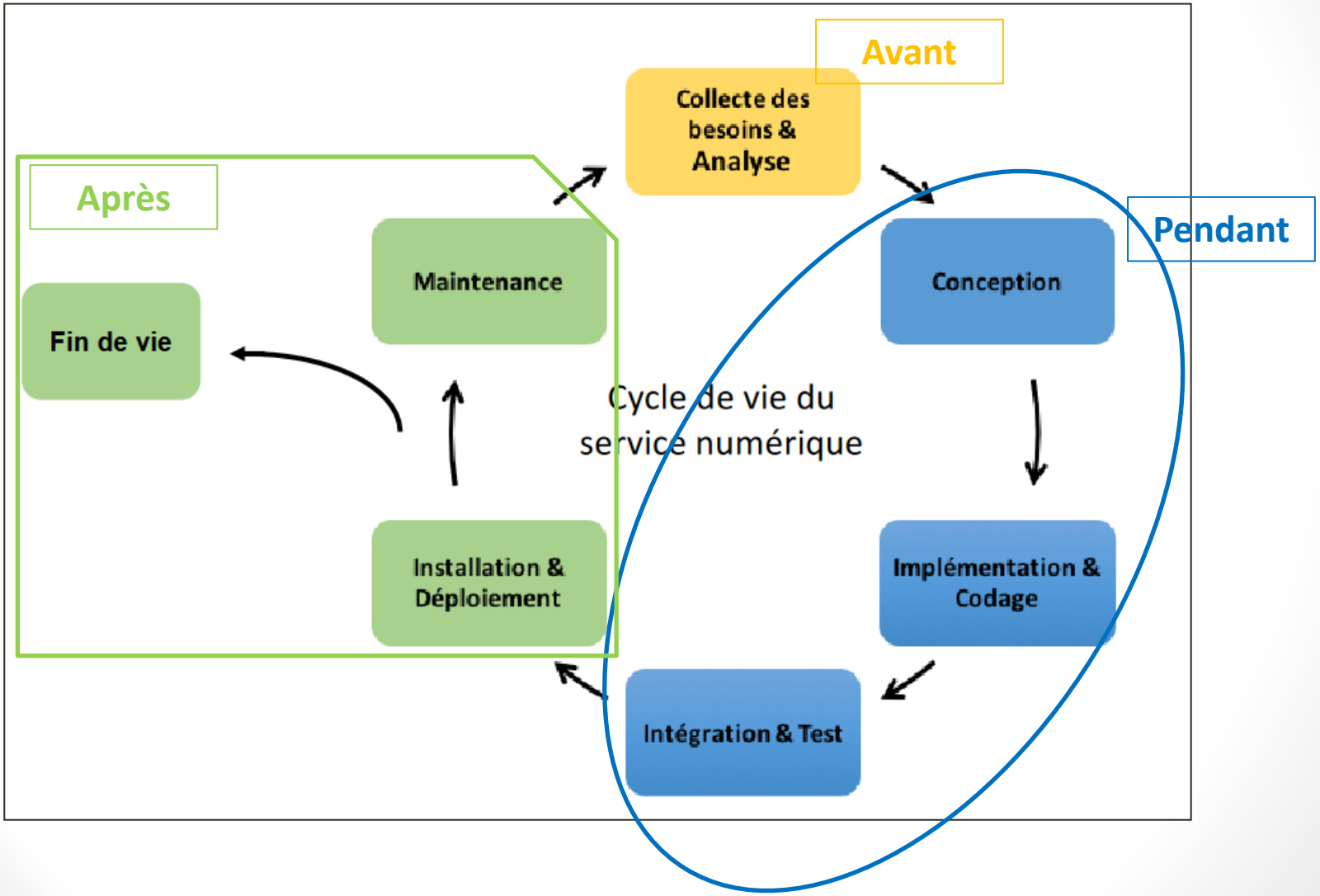
Éco-concevoir parce que…

- le logiciel
  - est au cœur du service numérique
  - conditionne le matériel nécessaire
  - Conditionne la durée de vie du matériel
- la fabrication du matériel est non négligeable d'un point de vue environnemental
- Le logiciel peut provoquer un renouvellement contraint du matériel
- Logiciel efficace/sobre :
  - Moins de besoin de puissance de calcul, mémoire et stockage,
  - Réduction du volume de matériels nécessaires au service numérique
  - Diminution de l'impact de la phase de fabrication des équipements
  - Diminution de l’impact de la consommation énergétique en fonctionnement dans une moindre mesure.

Cékoitess, l'éco-conception de services numériques ?

- Intégrer des contraintes environnementales dans tous les processus de développement (réduire sur tout le cycle de vie)
- Les impacts peuvent avoir lieu
  - Avant développement (définition des besoins, analyse)
  - Pendant le développement (conception, développement, tests, mise en production)
  - Après le développement (exploitation, maintenance, fin de vie)
- Plus la prise en compte est précoce dans le CV, plus l'effet est important.
- Possible quel que soit le cycle de développement choisi

Cycle de vie du logiciel

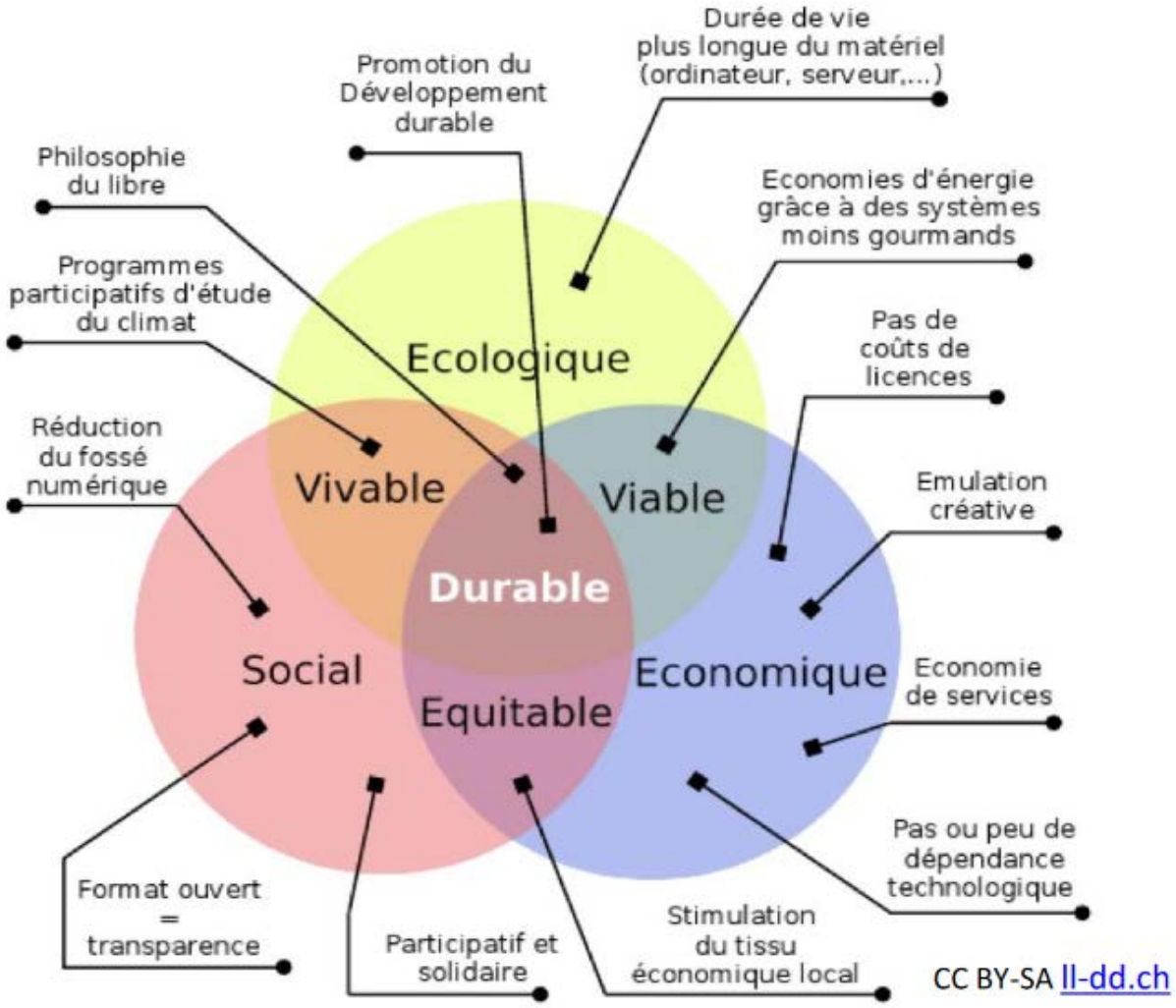


Quelques grands principes

- Simplicité : éviter les usines à gaz
  - 70 % des fonctionnalités demandées par l'utilisateur ne sont jamais ou rarement utilisées
  - Simplifier l'interface
  - Éviter les obésiciels !
- Frugalité et sobriété
  - Limiter le nombre et la taille des éléments (*e.g.* : images)
  - Exemple web : réduire les fonctionnalités et le graphisme
- Pertinence = utilité X rapidité X accessibilité
  - Utilité : répondre à l'attente de l'utilisateur
  - Rapidité : temps de réponse pour l'utilisateur
  - Accessibilité : penser au handicap
- Durabilité :
  - réutiliser l'existant pour ne pas dupliquer
  - Contribuer pour la communauté

Avant

Favoriser le libre



À lire : https://ll-dd.ch/

Planifier la gestion du logiciel

Avant

Accroître la durée de vie

- Utiliser un plan de gestion du logiciel (SMP, Software Management Plan) pour regrouper et mettre à jour les données du logiciel

➢ Produit modifiable et réutilisable facilement

- exemples : Opidor (https://dmp.opidor.fr/), Presoft (http://www.france-grilles.fr/wp-content/uploads/2018/04/ModeleSMP_PRESOFTV3.2.pdf)

Réfléchir au déploiement

Avant

S’adapter au mieux au contexte

- En fonction des caractéristiques (lieu, capacités…) des plateformes disponibles
  - Ordinateur, serveur local, plateforme embarquée
  - Plateforme spécialisée (cluster de virtualisation, cluster spécialisé HPC, GPU, ARM64, etc)
  - Offre de service, de site, ou de tutelles, cloud public
- En fonction des contraintes du service
  - Langages supportés, communications spécialisées (infiniband, Omnipath, etc)
  - Goulots d’étranglement : accès disque, réseau, transferts mémoire
  - Pérennité, portabilité, sécurité, coûts à long terme, maintenance, temps de retour
  - Lieu géographique des usages, tutelles commanditaires du service

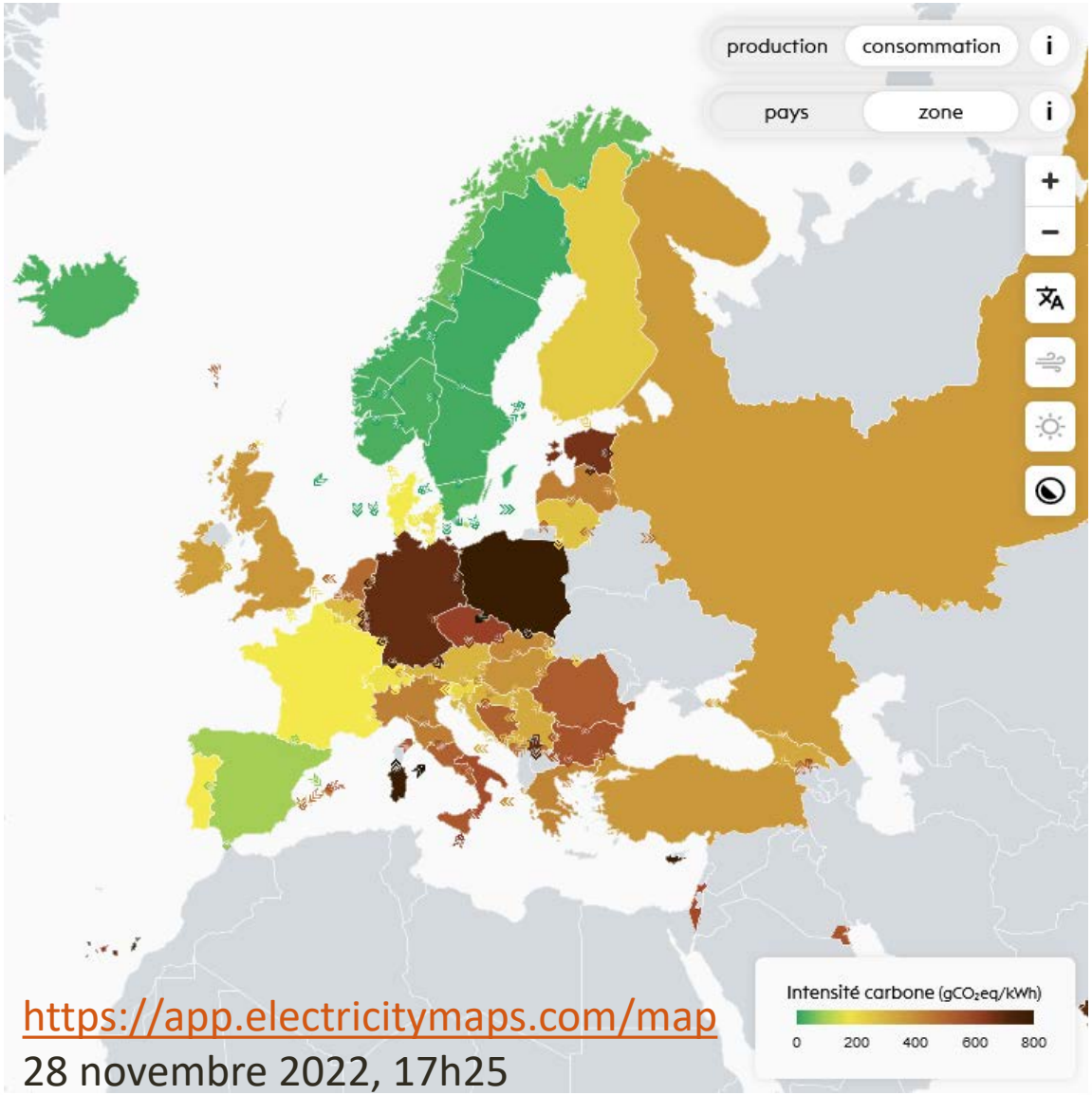
Déploiement : recommandations

Avant

- S’appuyer sur les services mutualisés (authentification, stockage, base de données…)
- Isoler les données sur des espaces de stockage distribués et sauvegardés (évite la duplication)
- Privilégier la virtualisation et réduire la taille des images (VM, container)
- Estimer au plus près les ressources nécessaires (vCPU, mémoire, stockage) pour éviter de surdimensionner
- Automatiser le déploiement des machines physiques et virtuelles (garantir la reproductibilité)

Avant

Dans quel pays ?



https://app.electricitymaps.com/map

28 novembre 2022, 17h25

Attention à la sécurité

Avant

- Gare aux failles de sécurité (indisponibilité de service, vol de données utilisateur, malwares, etc.)
- Voir les recommandations de l'Agence Nationale de la Sécurité des Systèmes d'Information
  - RUST : https://www.ssi.gouv.fr/guide/regles-de-programmation-pour-le-developpement-dapplications-securisees-en-rust/
  - Java : https://www.ssi.gouv.fr/agence/publication/securite-et-langage-java/
- Rester vigilant·e sur l'impact des mécanismes de sécurité considérés (augmentation de la consommation de ressources ou du nombre d'équipements)
- Prendre en compte la criticité des services : ne pas sur-sécuriser inutilement

Choisir son langage/sa pile logicielle

Avant

- Faire des compromis
- Importance des compilateurs et des facteurs humains (optimisation de code, expertise, test et mesure, …)

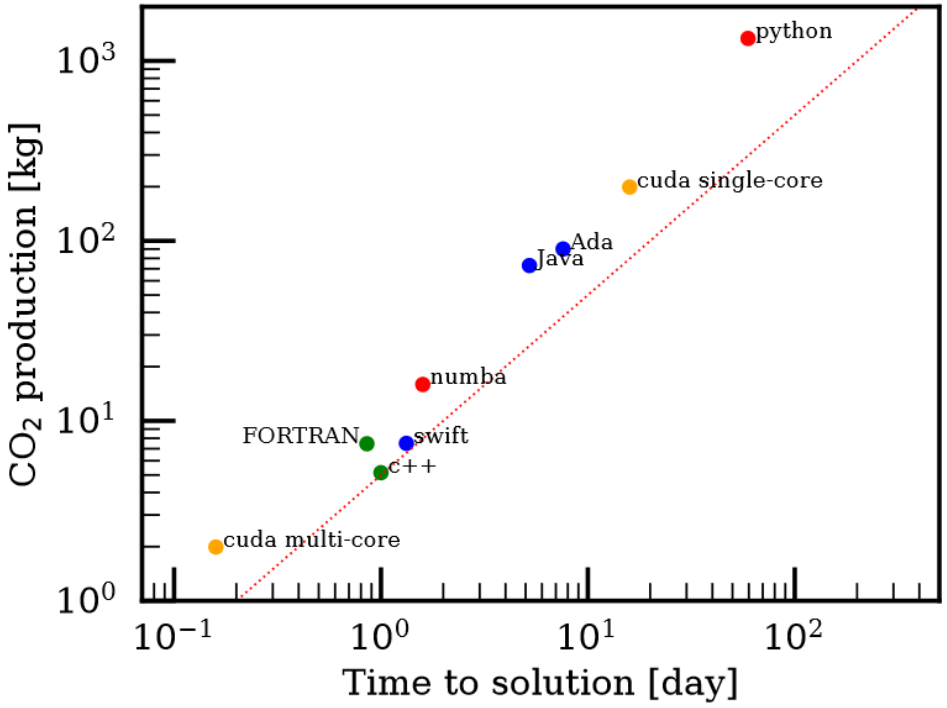


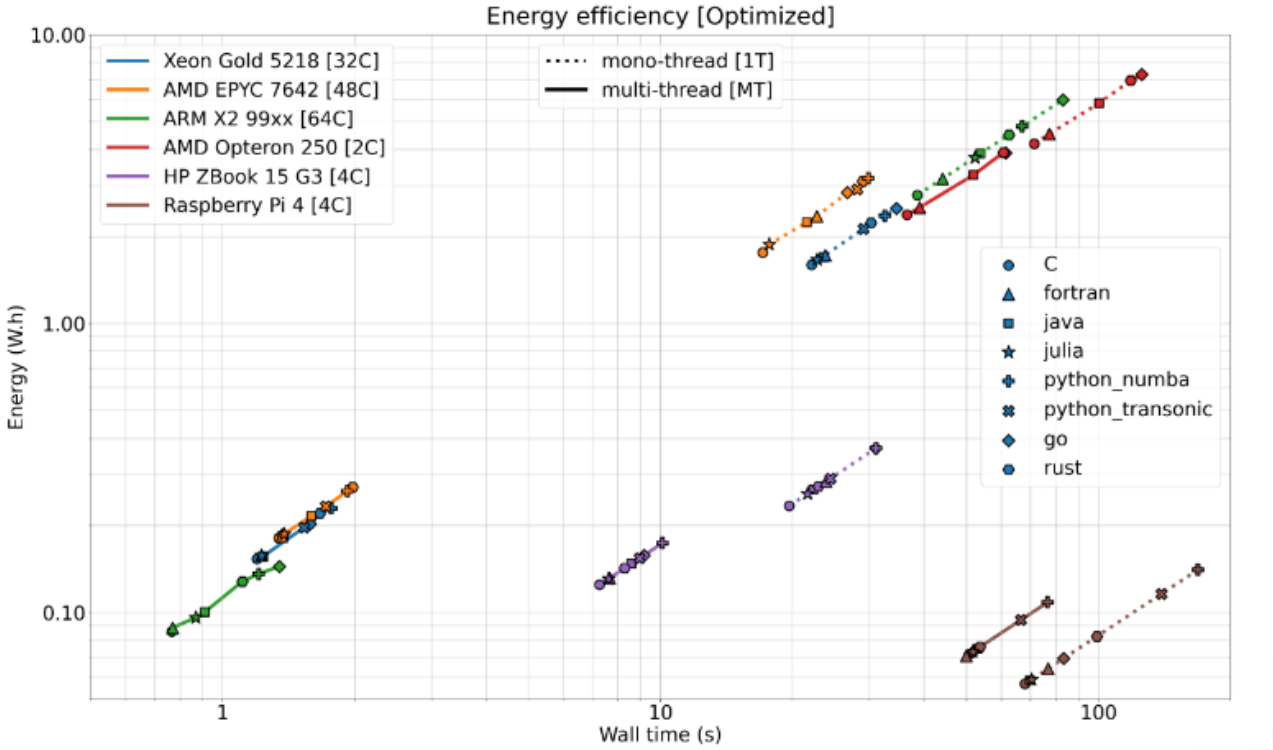
Figure 3: **Here we used the direct N-body code from [23] to measure execution speed and the relative energy efficiency for each programming language from table 3 of [22]. The dotted red curve gives a linear relation between the time-to-solution and carbon footprint ($\sim 5\,\mathrm{kg\,CO_2/day}$). The calculations were performed on a 2.7GHz Intel Xeon E-2176M CPU and NVIDIA Tesla P100 GPU.**

Source : https://arxiv.org/pdf/2009.11295.pdf

Python à la loupe

Avant

Écart de presque 100x en efficacité selon l'environnement d'exécution (cpython, numba, pythran, pypy) et l'optimisation du code dans ce cas d'utilisation (performance-driven development)



Source : Bonamy, Bourgès et Lefèvre
https://hal.archives-ouvertes.fr/hal-03607468/document

Concrètement, comment choisir ?

Avant

- Langages compilés (natifs ou interprétés, mais optimisés) pour les traitements lourds haute performance ou temps réel
- Langages faciles d'accès (interprétés) pour les traitements moins contraints pour faciliter la maintenance, le réemploi
  - durabilité
- On peut être polyglotte : toutes les briques ne doivent pas forcément être dans le même langage.

Question des données

Avant

- Stocker en ASCII ou en binaire ?
  - Le format texte (ASCII brut, XML, YAML...) nécessite généralement plus de place pour stocker la même information
  - Peut s'inverser si on versionne les fichiers (stockages successifs *vs* stockage différentiel)
  - Risque d'obsolescence si formats binaires non ouverts
- Volumétrie
  - Privilégier le traitement au plus près des données (évite les transferts de données sur les postes utilisateurs et le besoin de machines puissantes)
  - Concevoir les traitements et transferts en tenant compte de la volumétrie
  - Minimiser la volumétrie du paquet logiciel (archive) et éliminer les dépendances superflues

Avant

Planifier la gestion des données

- Durabilité
- Diminution des développements redondants
- Suivre les principes du FAIR :
  - Findable
  - Accessible
  - Interoperable
  - Reusable
  - https://www.ouvrirlascience.fr/portail-web-de-leosc/
- Le FAIR fonctionne aussi pour le logiciel :
  - https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg
- Penser au RGPD (Règlement Général sur la Protection des Données) https://www.cnil.fr/fr/comprendre-le-rgpd

Pendant

Analyser le code

Identifier les enjeux *a priori*

- Analyses statiques pour déterminer la difficulté de maintenance (exemples d'outils : Sonarqube, Codacy)
- Analyses dynamiques pour quantifier l'usage des ressources (exemples d'outils : gcov, gprof, perf, valgrind)

➢ Ce sont de bonnes pratiques de base

➢ Permet de réduire l'impact d'un service numérique

Pendant

Mesurer les performances

Identifier en fonctionnement

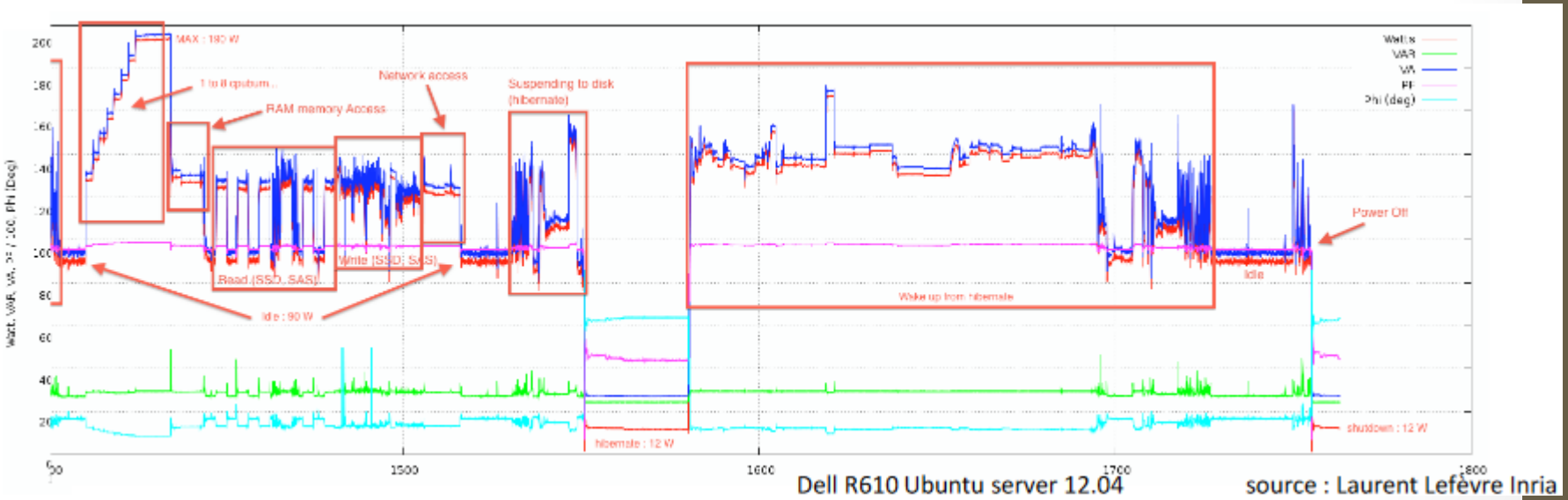
- temps de chargement
- temps d'exécution
- consommation électrique
- consommation des protocoles réseau (par exemple dans l'IoT)
- taille des données
- nombre de requêtes
- performance web
- mesure émissions carbone

Exemples d'outils : Firefox web tools (trafic, requêtes, JavaScript), Apache JMeter (scénarios web), ecoindex (web), profiler intégré et adapté au langage, CodeCarbon, Green Algorithms, KDE eco)

Pendant

Bonus consommation d'énergie

- Outils : cf. début du chapitre
- Permet d'identifier les différentes phases intensives du logiciel



Dell R610 Ubuntu server 12.04

source : Laurent Lefèvre Inria

Pendant

Optimisation : attention à l'effet rebond !

- Optimiser un logiciel => lancer d'avantage d'opérations ou traiter davantage de données
- L'optimisation devrait servir simplement à
  - réduire la consommation énergétique
  - réduire les ressources
  - si possible arriver plus vite au résultat
- Chaque exécution a un impact !
- Il est primordial de n'optimiser que ce qui a le plus d'impact (Loi de Pareto).
- Ne pas négliger la qualité des tests dans le processus d'optimisation (régressions)

Pendant

Amélioration : attention !

- L'amélioration est un problème multifactoriel
  - Architecture
  - Ressources processeur
  - Mémoire
  - Stockage
  - Trafic réseau
  - ➢ Attention à ne pas nuire à une ressource en en améliorant une autre !
- Exemples
  - système de cache => accélération des traitements MAIS impacts sur : mémoire, stockage et complexité du logiciel
  - compression => réduction trafic réseau MAIS augmente l'utilisation du CPU

Gestion de versions et intégration continue

Pendant

- Gestion de version
  - Éviter ou limiter le stockage de paquets binaires et de jeux de données non indispensables
  - Ne pas versionner les produits de compilation ni les fichiers de sortie
- Intégration continue
  - Réfléchir
  - Choisir un docker de taille minimum
  - N'activer que sur certaines branches
  - Envisager une exécution programmée
  - Surveiller la durée et le nombre des jobs, la taille des artefacts, le trafic réseau
  - Privilégier les forges mutualisées

Après

Déploiement et sobriété numérique

- Privilégier un hébergement
  - mutualisé
  - labélisé CoC (https://ecoinfo.cnrs.fr/2020/05/19/guide-des-bonnes-pratiques-du-code-de-conduite-europeen-sur-les-datacentres/)
  - proche des données et des utilisateurs
- Privilégier la virtualisation (moins de serveurs) en minimisant l'empreinte mémoire et disque (taille des machines, des containers), sauf cas particuliers (HPC)
- Attention à certains effets rebond
  - Éviter de multiplier les VM
  - Éviter de multiplier les instances du service

Après

En production

- Exploiter la supervision (et les alertes) pour observer
  - les pics CPU
  - les ressources utilisées (disque, réseau)
  - la consommation électrique
- Modifier le service pour l'adapter en fonction de l'usage observé (amélioration continue en fonction des usages)
- Réduire les fréquences et volumes des sauvegardes
- Adopter la déduplication

Exemples d'outils de supervision : top, vmstast, zabbix, scalasca, nagios, prometheus, grafana

Conserver l'efficacité énergétique en production

Après

- Éteindre les machines
  - Le logiciel influence la consommation dynamique des machines
  - Peu de proportionnalité énergétique
  - La consommation statique d'un serveur ou d'un équipement réseau est importante
  - Un serveur inoccupé peut consommer 50% de son budget électrique
- Mettre en place un système de veille
  - Éteindre les VM qui ont une empreinte mémoire et CPU
  - Limiter le nombre de services déployés
  - Favoriser l'extinctions d'hôtes et de ressources (cœurs de calcul, systèmes de stockage) inutilisés

Après

Distribution et maintenance

Favoriser la durabilité et la simplicité

- Diffusion
  - Dépôt à un endroit unique, facilement accessible (forge)
  - Préférer la publication Open Source
    - https://hal.archives-ouvertes.fr/hal-02399517/file/20191202_plaquette_pi_licences_V1.1.pdf
    - https://www.etalab.gouv.fr/accompagnement-logiciels-libres/
- Gestion des mises à jour
  - Réduire la taille des produits logiciels
  - Rationaliser le nombre et la fréquence des produits logiciels
  - Favoriser la rétrocompatibilié
  - Éviter l'ajout constant de fonctionnalités
  - Définir clairement le workflow des mises à jour
  - Support de Long Terme : https://fr.wikipedia.org/wiki/Long-term_support

Après

Fin de vie du logiciel 1/2

- Archiver en favorisant la durabilité
  - Déclarer le code source auprès de Software Heritage https://www.softwareheritage.org/save-and-reference-research-software/?lang=fr
  - Sauvegarder sur un stockage froid les données essentielles (valeur ou obligation juridique)
- Mettre hors-service en évitant les « zombies »
  - Utilisateurs : prévenir et s'assurer que les solutions de repli sont correctes
  - Documentation : supprimer les données et signaler la fin de vie dans l'espace de documentation
  - Outils de support : mettre à jour les configurations pour supprimer les files actives et nettoyer les données
  - Logiciel : supprimer l'installation sur toutes les instances

Après

Fin de vie du logiciel 2/2

- Mettre hors-service en évitant les « zombies »
  - Données : supprimer les données stockées après archivage
  - Services nécessaires : arrêter et désinstaller les services qui n'étaient utiles que pour l'exécution du logiciel (Apache, Nginx, mysql, etc.)
  - Machines virtuelles et containers : arrêter ce qui n'était pas mutualisé, supprimer leur image… permet parfois d'éteindre l'infrastructure physique !
  - Sauvegardes : toutes les supprimer
  - Supervision : libérer les outils de supervision et supprimer les traces d'exécution
  - Dépendances mutualisées : supprimer les éventuelles dépendances mutualisées (bibliothèques logicielles, assets JS ou CSS, etc.) ne servant pas à d'autres services
  - Cartographie applicative : supprimer les parties concernant le logiciel

Quelques usages spécifiques

Éco-concevoir le calcul scientifique

Avant

- Choisir la licence (ouverte)
- Choisir la technologie et le langage
- Choisir l'infrastructure adaptée pour chaque phase
- Réfléchir à la gestion des données
- Concevoir le workflow
- Alerter les partenaires sur l'impact environnemental du numérique

Début

- Créer un dépôt
- Créer les fichiers Authors, License, Readme
- Considérer les formats ouverts pour les sorties
- Mise en place de l'intégration continue (sans excès)
- Prévoir documentation et mise à jour
- Prévoir le redémarrage

Pendant

- Bien choisir les bibliothèques
- Diffuser, partager dès que possible
- Reproductibilité !

En production

- Chercher les points chauds
- Attention au poids des E/S
- Analyser l'extensibilité du code
- Se faire aider pour optimiser (attention rebond)
- Sensibiliser à l'impact des calculs
- Chercher les compromis impact/finesse/temps de réponse
- Réfléchir à l'intérêt des calculs
- Supprimer les données inutiles

Fin de vie

- Conserver le minimum
- Diffuser le code en précisant la fin de maintenance

Éco-concevoir du code GPU et/ou IA

Avant

- Attention au poids de l'apprentissage pour l'IA
- Considérer les alternatives à l'IA
- Sensibiliser les utilisateurs au coût des calculs, y compris phase d'apprentissage
- Choisir la technologie et les contraintes en fonction des besoins

Pendant

- Choisir les bons frameworks pour coder
- Ne pas développer ce qui existe déjà : utiliser des bibliothèques éprouvées de haut niveau
- Clé des accélérateurs
  - uniformiser (pas de conditions, pas d'hétérogénéité entre cœurs)
  - Maximiser les traitements sur l'accélérateur (GPU) => éviter les transferts mémoire
- Faire du code portable et performant
- Attention à l'impact des E/S

En production

- Sensibiliser les utilisateurs au coût des calculs, en incluant le coût de la fabrication des GPU (coût consolidé)
- Sondes logicielles (3 bibliothèques Python) Experiment Impact Tracker, Carbon Tracker et Code Carbon)

Éco-concevoir un site web

Avant

- Lire et assimiler le Référentiel Général d'Ecoconception de Services Numériques
- Gains
  - économiser les fonctionnalités
  - simplifier l'interface utilisateur (sobriété)
  - Limiter les éléments générant du trafic et du CPU sur les terminaux (vidéos, les traceurs, jeux de données, graphiques)
- Éliminer les contenus non essentiels
- Prévoir un usage sur petit écran
- Attention à la sécurité

Pendant

- Piloter la qualité de développement *via* les indicateurs d'impacts environnementaux des terminaux
- Idem en intégration continue
- Planifier la gestion des données au plus proche
- Penser FAIR
- Compatibilité et maintenance à long terme (technos pérennes)
- Limiter les accès réseau et transferts de données (ok faible débit)
- Code : Keep It Simple and Stupid

En production

- Choisir un hébergeur labellisé CoC, au plus proche des utilisateurs
- Politique de sauvegarde des données sobre
- Supervision du service (CPU, E/S) et des journaux
- Anticiper les usages (attention aux comportements induits)
- Adapter pour réduire les impacts négatifs

Résilience et sobriété

- Assurer la résilience en sécurisant
  - Prendre en compte les failles de sécurité potentielles sur tout le cycle de vie
  - Rester vigilant sur l'impact des mécanismes de sécurité mis en œuvre (augmentation de consommation de ressources ou nombre d'équipements)
- Low-Techs
  - Technologies peu consommatrices en énergie
  - Résistance à la panne
  - CollapseOS : système fonctionnant sur des microcontrôleurs http://collapseos.org/
  - Lowtech Magazine : hébergement d'un site web uniquement visible si la production d'énergie solaire est suffisante https://solar.lowtechmagazine.com/fr/about.html