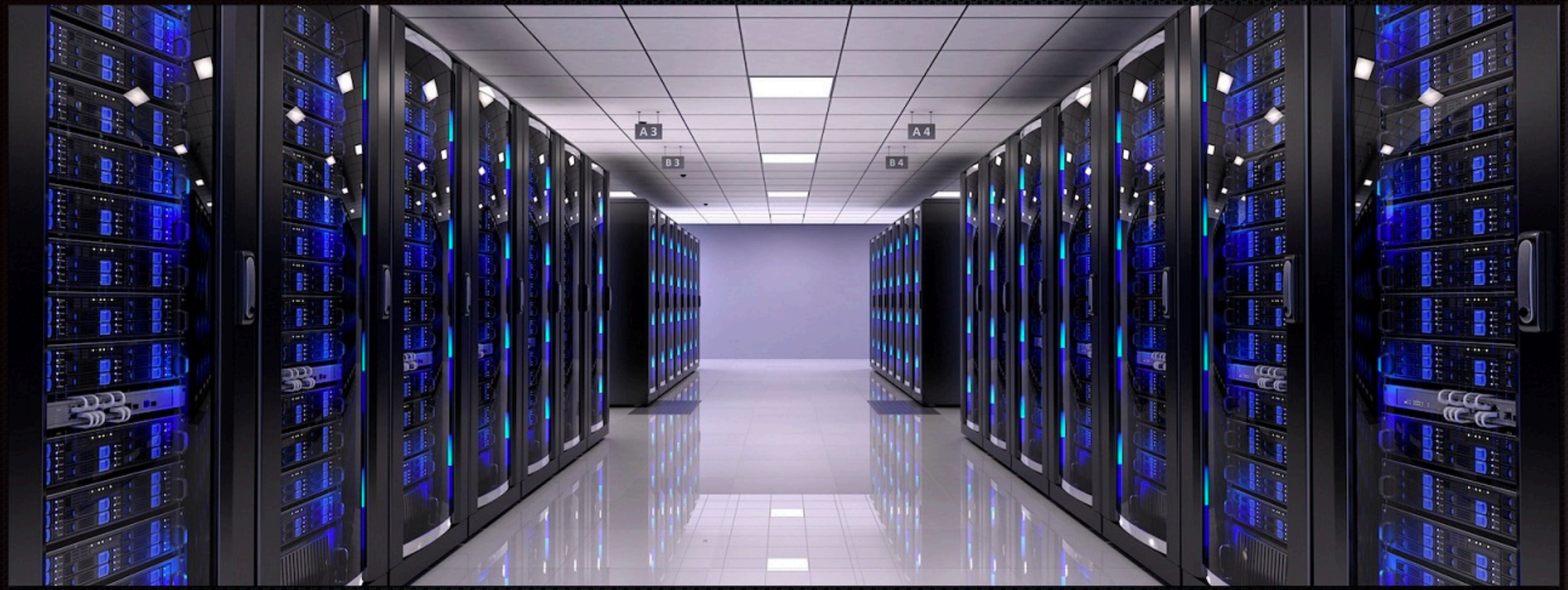


Maitrise de l’impact environnemental du numérique et des systèmes d’informations
Focus sur les problématiques d’éco-conception logicielle

Séminaire CRI du 21 juin 2021

Renaud Pawlak
CINCHEO, coordinateur de la R&D EASYTEAM (groupe Constellation) sur le bas carbone

Renaud Pawlak

- Ex-chercheur INRIA, responsable du département informatique de l'ISEP, ex-directeur du lab Mantu (IA)
- Co-fondateur de ID Capture : 10 ans de développement de services numériques pour l'industrie de la construction
- Créateur de CINCHEO, R&D développement logiciel responsable (JSweet, SweetHome3D, INI, DLite…)
- Coordinateur de la R&D de EASYTEAM (groupe Constellation) sur la maîtrise de l'empreinte environnementale du numérique

Programme de R&D EASYTEM

- Axe 1 : mesures et modèles d'impacts
- Axe 2 : conception éco-responsable
- Axe 3 : cartographie et aide à la décision (ACV conséquentielle)
- Plateforme collaborative ouverte pour les modèles d'impacts

PowerAPI

Inria

Cloud Carbon Footprint

+ Offre de formation

Plan

- Pourquoi s'intéresser à l'empreinte environnementale du numérique ?
- Introduction à l'éco-conception logicielle
- Efficacité énergétique
- Autres pistes de réflexions (gestions des dépendances, obsolescence des environnements, compilation)

Pourquoi s’intéresser à l’empreinte environnementale du numérique ?

Motivations

Impacts du numérique

- Aujourd'hui: le numérique représente 4% des émissions mondiales de GES (équivalent au secteur de l'aviation civile)
- Projections 2025: 8% (= voiture individuelle)
  - Progression exponentielle (Big Data, BlockChain, Machine Learning, …)
- Effets indirects de second ordre et multi-sectoriels (par exemple : changement de comportements lié au télé-travail)

Législations en cours autour de la "taxe carbone"
Accords de Paris => -50% d'ici 2030

Notion d'impact : effets multiples

- Effets de premier ordre (effets directs, cycle de vie, effets de substitution)
  - *Exemple: remplacement des appareils photos par des iPhones*
- Effet de second ordre (effets rebonds, paradoxe de Jevons)
  - *Exemple: taille des écrans LCD*
- Les impacts sont rarement cantonnés à un secteur
  - *Exemple : un service de e-commerce aura des effets sur les secteurs du recyclage, de l'emballage, du transport, et de la construction (stockage)*
  - *Exemple : le télé-travail au des impacts sur les transports et sur la construction (et donc sur l'énergie pour le chauffage, c.f. la crise sanitaire récente)*

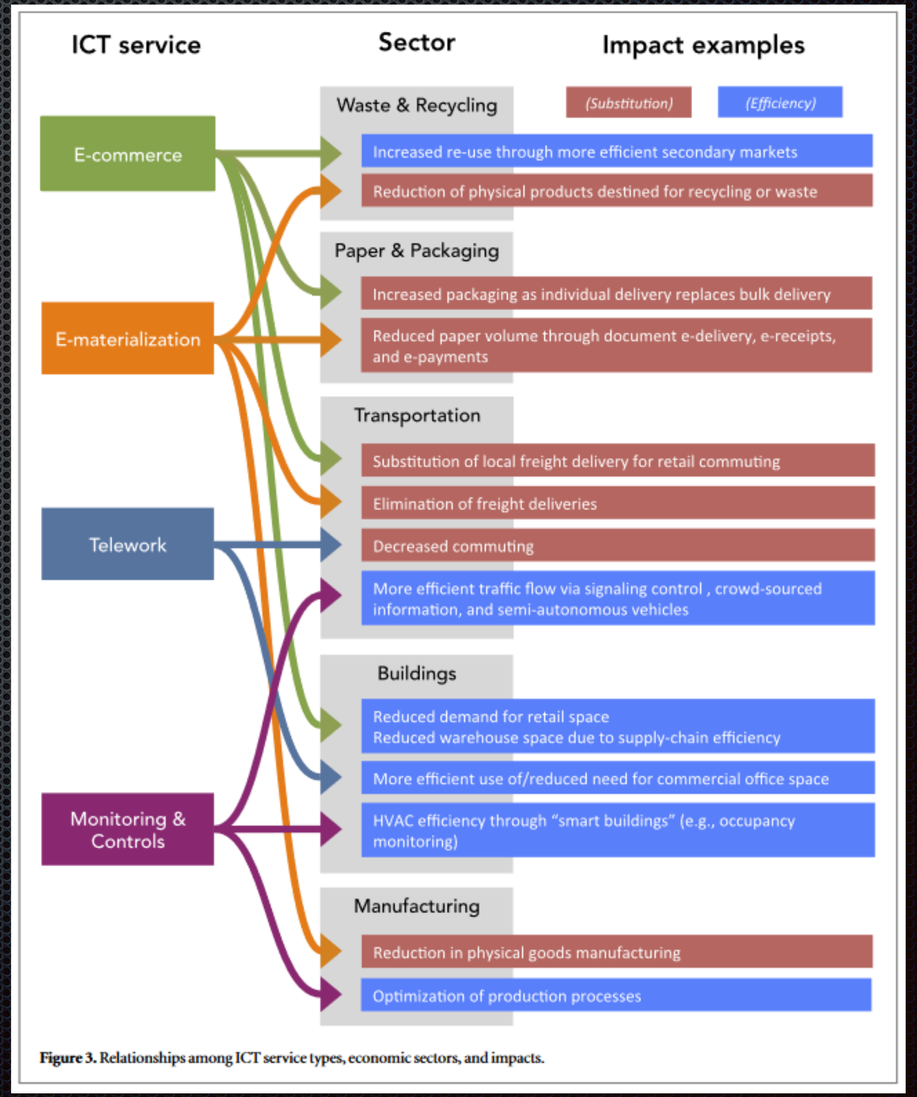


Figure 3. Relationships among ICT service types, economic sectors, and impacts.

Introduction à l'éco-conception logicielle

Définitions, enjeux et leviers

Définition (générale) de l'éco-conception

- L'Agence de la transition écologique (ADEME) définit l'éco-conception comme « une démarche préventive qui se caractérise par la prise en compte de l'environnement lors de la phase de conception ou d'amélioration d'un produit. L'objectif de cette démarche est d'améliorer la qualité écologique du produit, c'est-à-dire réduire ses impacts négatifs sur l'environnement tout au long de son cycle de vie, tout en conservant sa qualité d'usage ».
- En d'autres termes, éco-concevoir c'est chercher à réduire la quantité des ressources informatiques utilisées sans renoncer à l'utilité des services rendus et aux bénéfices associés.
- Pour aller plus loin, on peut essayer aussi de prendre en compte les bénéfices indirects et d'anticiper les effets rebond, ce qui nécessite une analyse systémique complexe.

* Référentiel Green-IT https://institutnr.org/wp-content/uploads/2020/06/2020-v3-65-bonnes-pratiques-greenit.pdf, #671, #626

Eco-conception logicielle : les enjeux

- L'impact le plus fort est lié aux terminaux utilisateurs (PC, smartphones), et en particulier à la fabrication
- Les enjeux court-terme de l'éco-conception logicielle sont donc :
  - De minimiser les effets d'obsolescence et de favoriser la durabilité du matériel (donc minimiser les resources CPU, mémoire, disque, etc.)
  - De minimiser la consommation énergétique des terminaux (mais aussi des data centers et des réseaux, le tiers de l'énergie consommée)

%	Energie	GES	Eau	Ressources[1]
Utilisateurs	64 %	84 %	91 %	79 %
Réseau	21 %	10 %	5 %	15 %
Centres informatiques[2]	15 %	6 %	4 %	6 %

%	Energie	GES	Eau	Ressources[1]
Fabrication	41 %	83 %	88 %	100 %
Utilisation	59 %	17 %	12 %	0 %

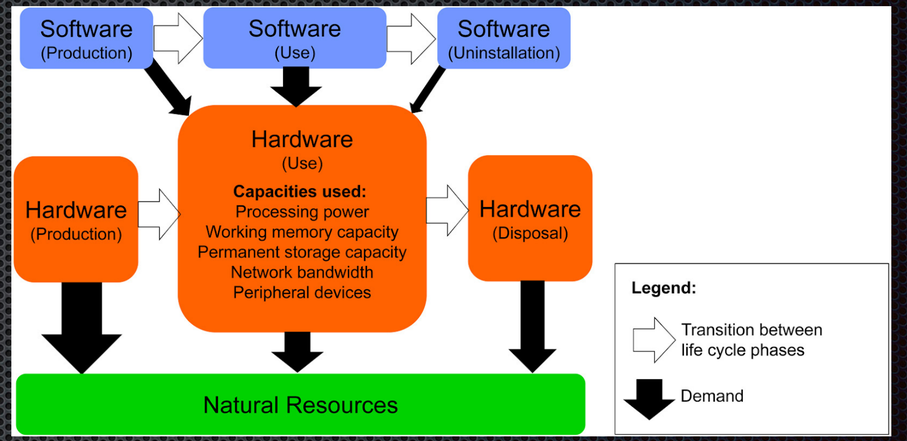
Répartition des impacts du numérique en France

https://www.greenit.fr/wp-content/uploads/2020/06/2020-06-iNum-etude-impacts-numerique-France-rapport.pdf

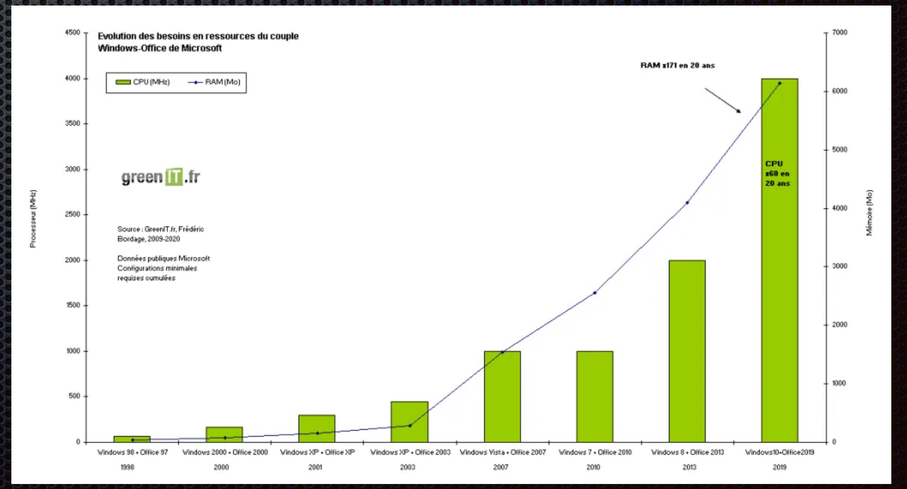
Des leviers importants à exploiter

- Le logiciel est en début de chaîne
- La loi de Wirth* nous donne un effet de levier important

* “Slowing down programs is much faster than speeding up computers”



Kern E, Hilty LM, Guldner A, Maksimov YV, Filler A, Gröger J, Naumann S. Sustainable software products—Towards assessment criteria for resource and energy efficiency. Future Generation Computer Systems. 2018 Sep 1;86:199-210.



Exemples

- Optimiser en fonction de l'usage (statistiques)
  - Exemple : l'étude de l'usage du moteur de recherche Bing a permis par exemple, de mettre en avant que 90% des personnes n'étudient que le top 20 des liens retournés. L'exploitation de cette statistique permet de réduire l'impact environnemental des serveurs de 80 % (*).
- Segmenter des données en fonction des usages
  - « La Deutsche Bahn démontré qu'il était possible de diviser par 1350 la quantité de ressources informatiques nécessaires pour trouver l'horaire d'un train, si on utilise uniquement les données nécessaires » (source : Frédéric Bordage, GreenIT (**)).
- Utiliser des "green patterns" (protocoles, algorithmes, etc.)
- Utiliser des outils de compilation / transpilation / analyse de code, etc.
  - Par exemple, en 2010, Facebook a divisé par deux le nombre de serveurs nécessaires à son fonctionnement en modifiant le code de ses services (compilation du code PHP de son site en C++). Le réseau social émet ainsi 2 fois moins de GES qu'auparavant et a évité la construction d'un nouveau data center qui lui aurait coûté environ 100 millions de dollars et aurait émis des dizaines de tonnes de GES inutilement.

* https://www.greenit.fr/2014/12/03/bing-20-des-resultats-sollicitent-80-des-ressources/

** http://videos.senat.fr/Datas/senat/portail/video.1504468_5e2f71cae6b58.table-ronde-relative-a-l-empreinte-carbone-du-numerique

Efficacité énergétique

Mesure et axes d’optimisation

Efficacité v.s. Performance

- La **performance** consiste à faire vite les tâches
- L'**efficacité** consiste à faire en priorité les tâches à forte valeur ajoutée
  - Exemple: répondre à 100 emails par jour v.s. répondre aux deux emails qui vont débloquer des situations critiques et vont me permettre de travailler sur d'autres sujets plus rentables
- L'efficacité est donc liée à un critère d'**utilité**
- Pour faire de la conception éco-responsable; il faut donc prendre en compte
  1. La consommation en énergie
  2. L'utilité **(sous contraintes QoS)** du service rendu

Efficacité Energétique = Utilité du service / Énergie utilisée

Exemple

- Serveur de génération de rapports
- Contrainte de QoS : chaque rapport doit être généré en moins d'une seconde
- 15 vs 25 utilisateurs simultanés pendant 1 heure d'utilisation
- Résultat : 25% plus efficace avec 25 utilisateurs

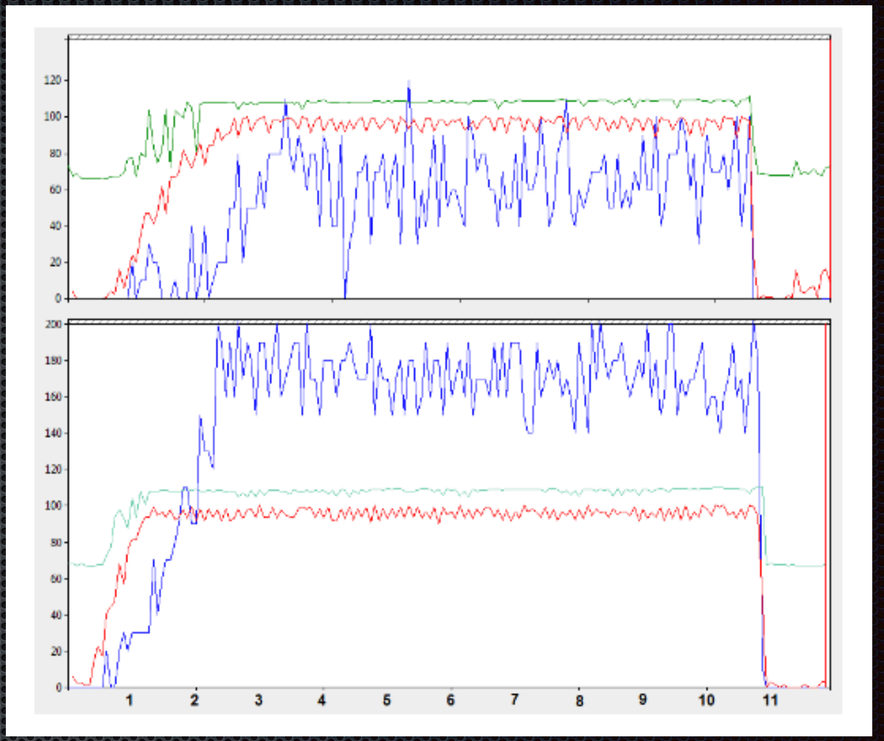


Table I
ENERGY PER USER

User Count	15 Users	25 Users
Energy Consumption	$7,3 \frac{Joules}{DeliveredChart}$	$5.5 \frac{Joules}{DeliveredChart}$
Average Response Time	0.5 seconds	0.8 seconds

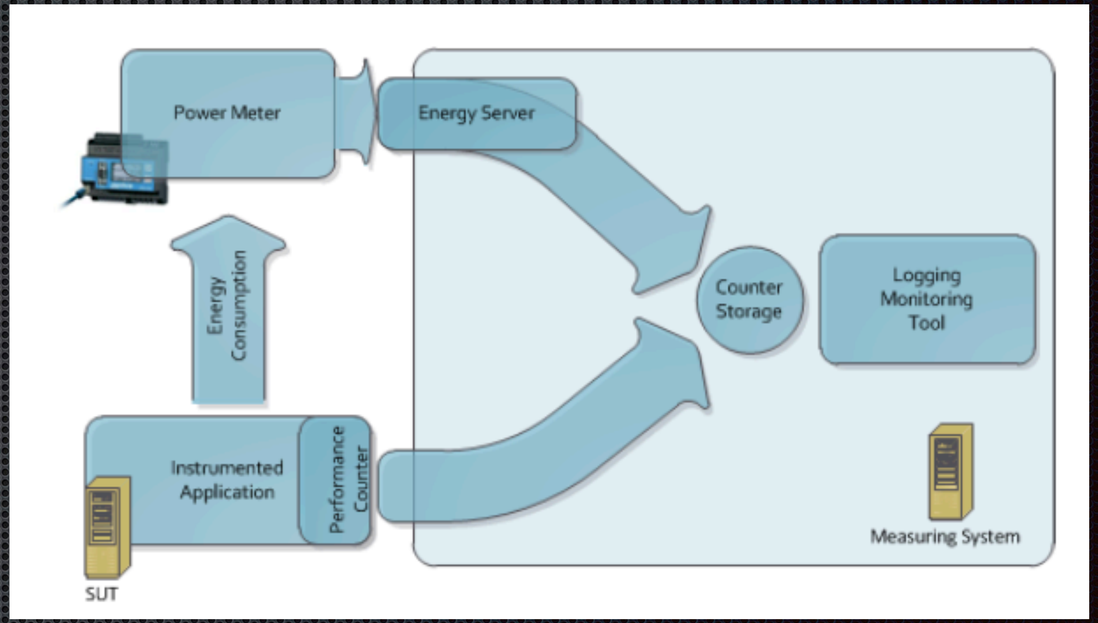
Source: How to Measure Energy-Efficiency of Software: Metrics and Measurement Results
Timo Johann, Markus Dick, Stefan Naumann, Eva Kern
Trier University of Applied Sciences, Germany

Mesure de l'énergie utilisée

- Avec un wattmètre
- Avec des sondes logicielles
  - PowerAPI (INRIA)
  - MC2 (CINCHEO)
  - cloudcarbonfootprint.org

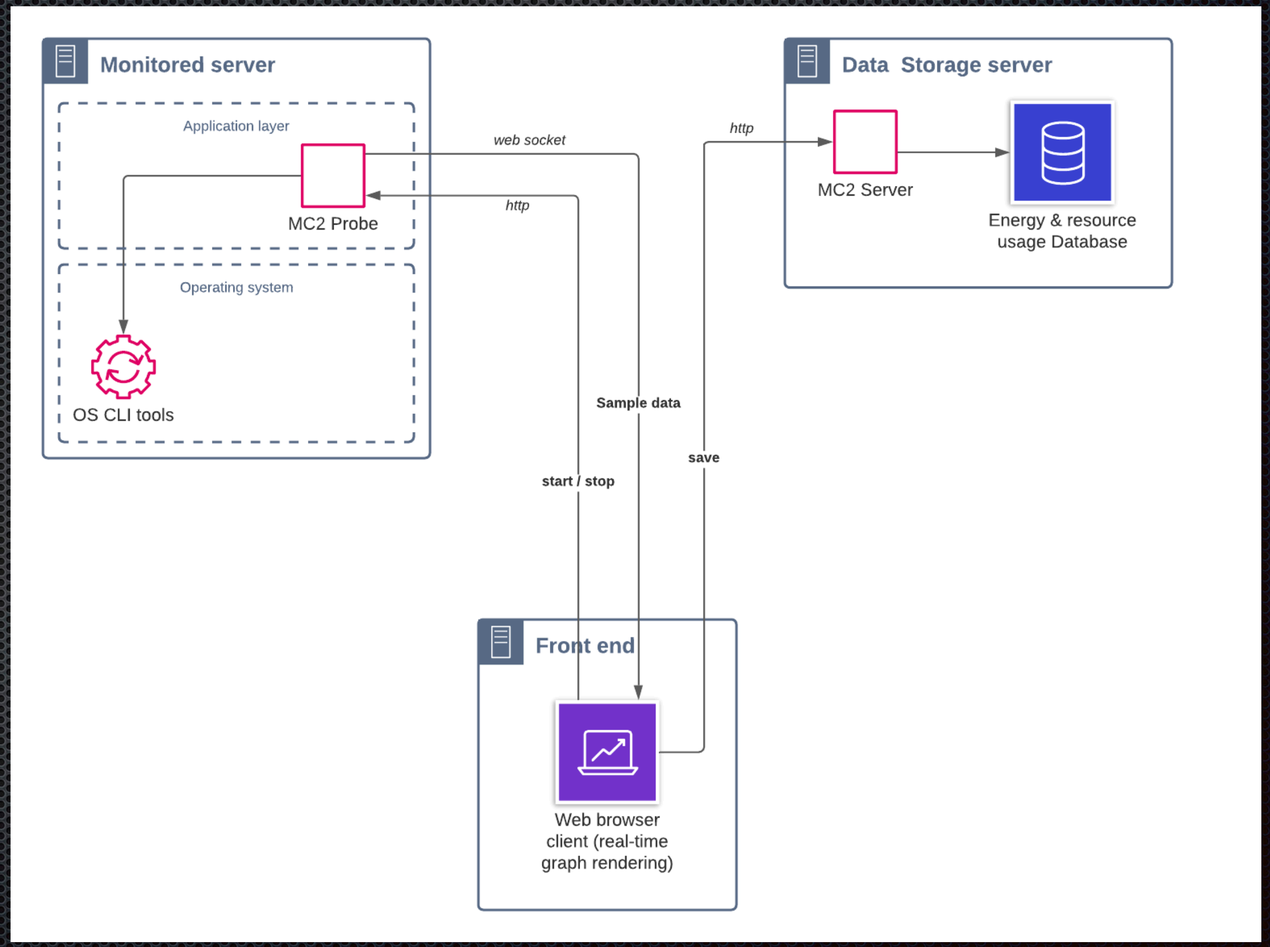
Monitoring et logging

- Les données de mesure de l'énergie doivent être corrélées au tâches effectuées (durée et performance d'exécution)
- Journaux systèmes et applications (outils de gestion de logs type Graylog), outils de monitoring type Zabbix
- Instrumentation de code pour augmenter la granularité des mesures (par exemple pour chaque fonction d'un programme ou d'une librairie)



MC2

- Monitoring de ressources implémenté en DLite (facilement extensible)
  - https://cincheo.com/2021/06/11/mc2-a-tool-to-remotely-monitor-computer-resources/
  - https://ui.dlite.io/
  - https://github.com/cincheo/dlite



Analyse de spectres de programmes

- Problématique
  - Soit un programme ou un service composé de N éléments (N instructions, N fonctions, N composants, N process, etc)
  - Comment trouver les composants les plus consommateurs de ressources pour les optimiser?
- L'analyse de spectre de programme est habituellement utilisée pour localiser des bugs, mais on peut l'utiliser pour localiser les points chauds
  - Le spectre d'un programme est la matrice résultante de l'exécution de M variantes du programme (on peut par exemple utiliser les tests)

Function	Statement number	Test Cases Fault1 S4:m=x;Fault2 S6:m=y; Fault3 S9:m=z;Fault4 S11:m=z;									
		T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
mid (x,y,z) { int m;	input	3, 3, 5	1, 2, 3	3, 2, 2	5, 5, 5	4, 4, 3	5, 3, 4	3, 2, 1	5, 4, 2	2, 1, 3	5, 2, 6
m=z;	S1	1	1	1	1	1	1	1	1	1	1
if(y<z)	S2	1	1	1	1	1	1	1	1	1	1
if(x<y)	S3	1	1				1			1	1
m=y;	S4		1								
else if(x<z)	S5	1					1			1	1
m=x;	S6	1								1	1
else	S7			1	1	1		1	1		
if(x>y)	S8			1	1	1		1	1		
m=y;	S9			1				1	1		
else if(x>z)	S10				1	1					
m=x;	S11					1					
print(m);	S12	1	1	1	1	1	1	1	1	1	1
}	Pass/Fail	P	F	P	P	F	P	F	F	F	F

N éléments impliqués

M traces mesurées

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \cdots & & & \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{pmatrix} \begin{pmatrix} \overline{E_1} \\ \overline{E_2} \\ \cdots \\ \overline{E_M} \end{pmatrix}$$

Energie consommée par élément

$$E_i = \frac{1}{n_i} \sum_{j=0}^{n_i} \frac{\overline{E}_j}{k_j}$$

n_i : nombre de traces dans lequel i est utilisé

$\overline{E}_j$: énergie moyenne consommée par la trace j

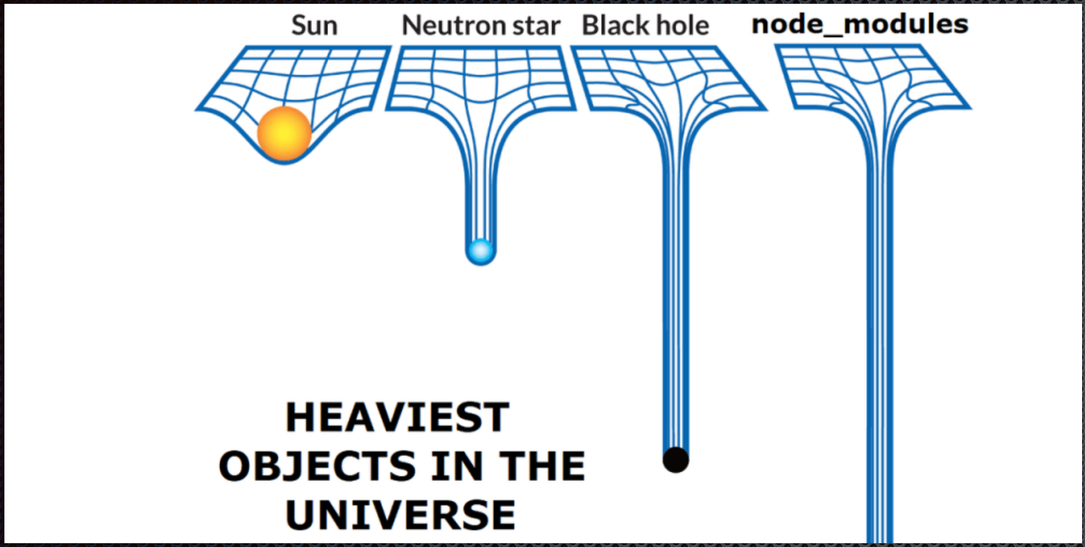
k_j : nombre d'éléments impliqués dans la trace j

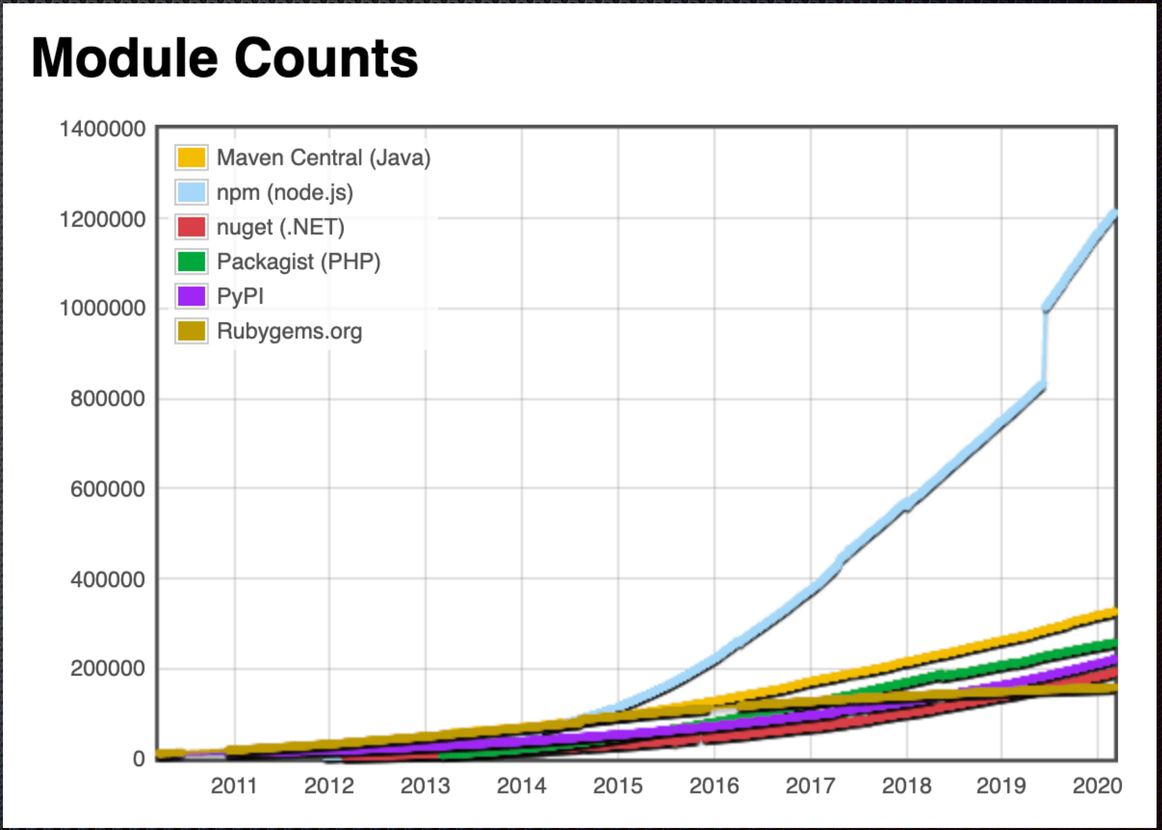
Source: Verdecchia, Roberto, et al. "Code-level energy hotspot localization via naive spectrum based testing." Advances and New Trends in Environmental Informatics. Springer, Cham, 2018. 111-130.

Autres pistes de réflexions pour l'éco-conception

Maîtrise des dépendances, modernisation des logiciels, compilation

Les gestionnaires de dépendances

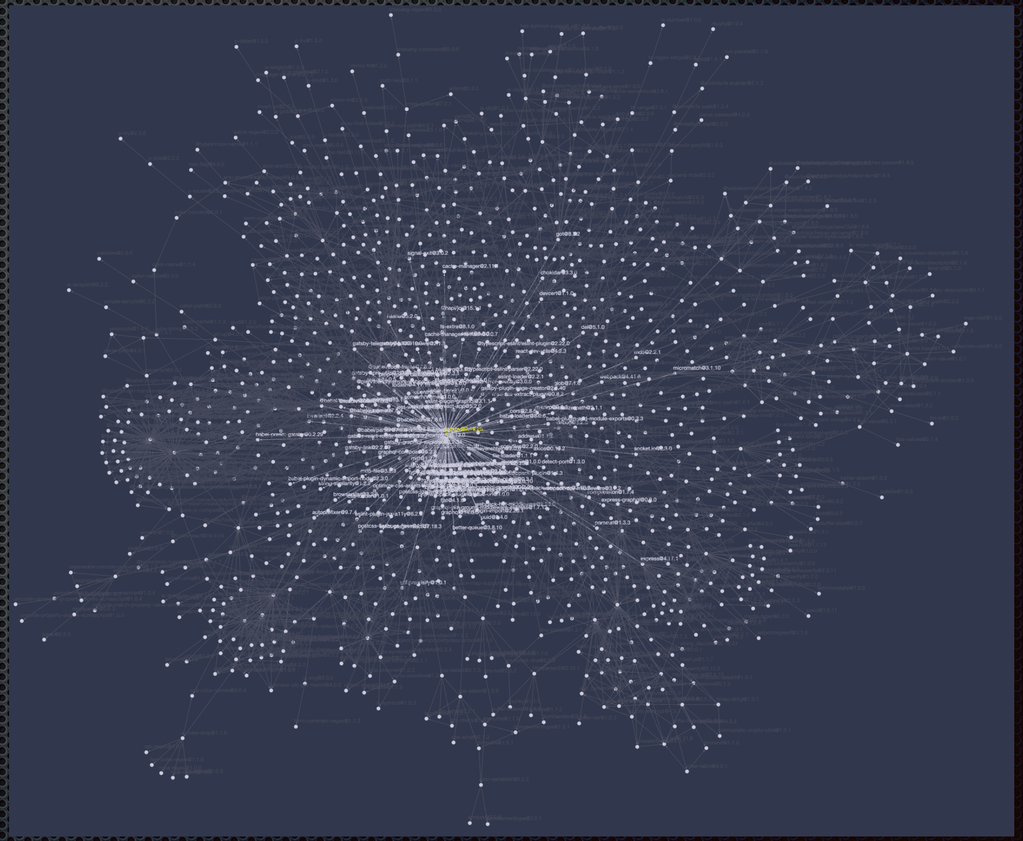




- Un cas d'école du paradoxe de Jevons !

Le cas du framework Gatsby

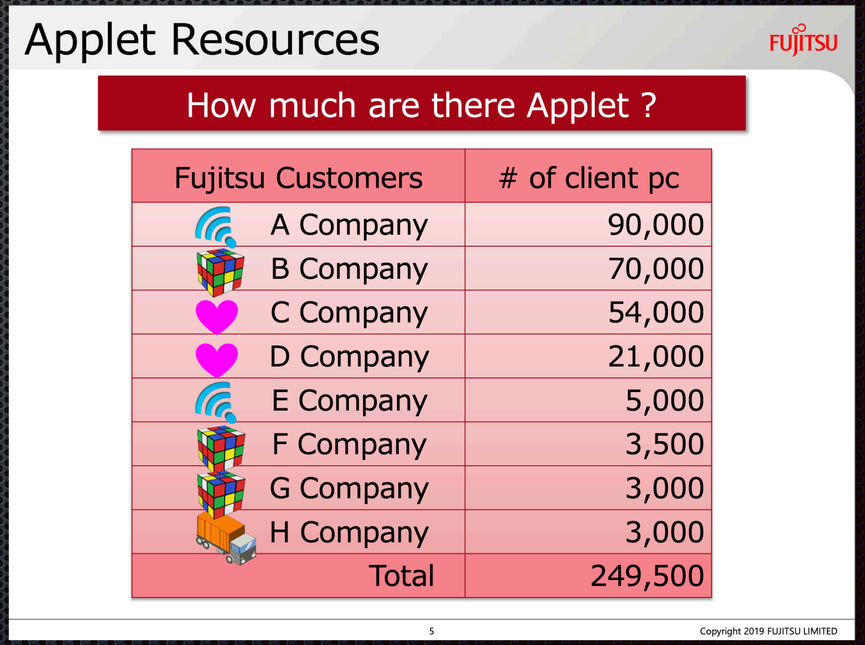
- Un framework front basé sur React.js
- 15K dépendances indirectes



https://npm.anvaka.com/#/view/2d/gatsby

Cas de la modernisation des logiciels

- Obsolescence des librairies et des supports d'exécution
- Par exemple, pour Fujistu, 249,500 Applets Java à maintenir dans des environnements obsolètes
- Réécriture, réécriture partielle, emulation ?



Applet Resources

FUJITSU

How much are there Applet ?

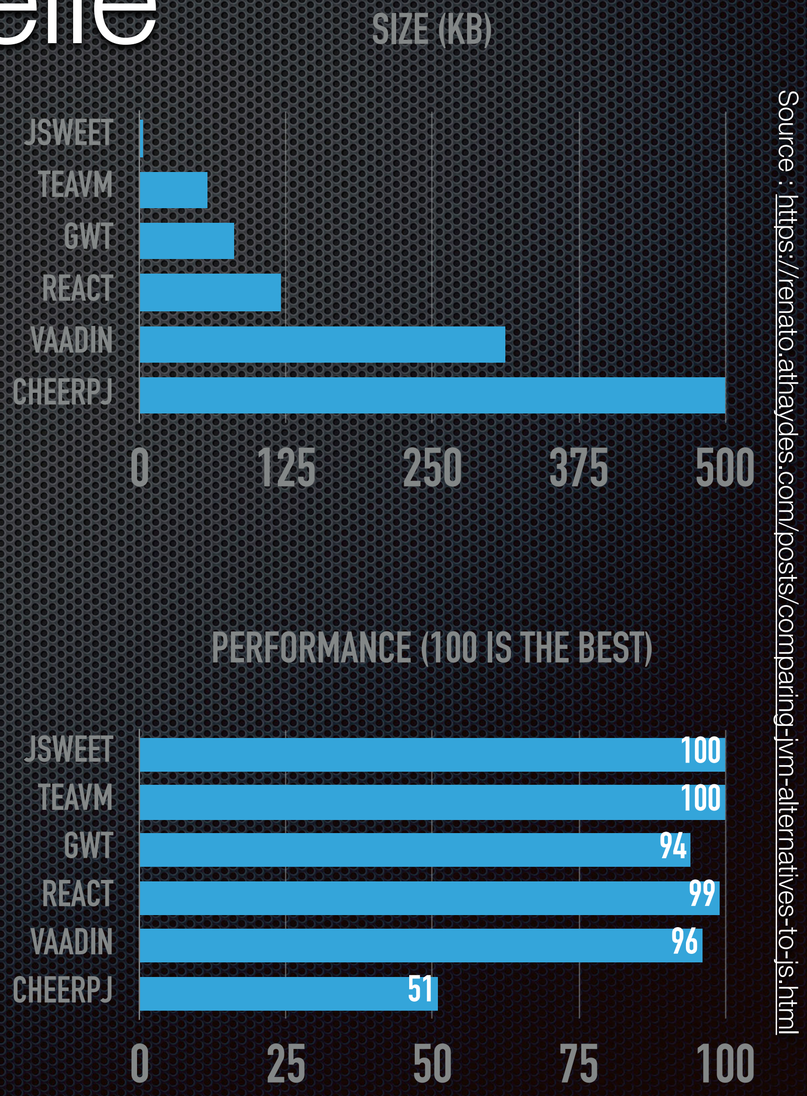
Fujitsu Customers	# of client pc
A Company	90,000
B Company	70,000
C Company	54,000
D Company	21,000
E Company	5,000
F Company	3,500
G Company	3,000
H Company	3,000
Total	249,500

5

https://jcp.org/aboutJava/communityprocess/ec-public/materials/2019-05-15/JCP_Fujitsu_20190515.pdf

L’approche de la transpilation ouverte pour la durabilité logicielle

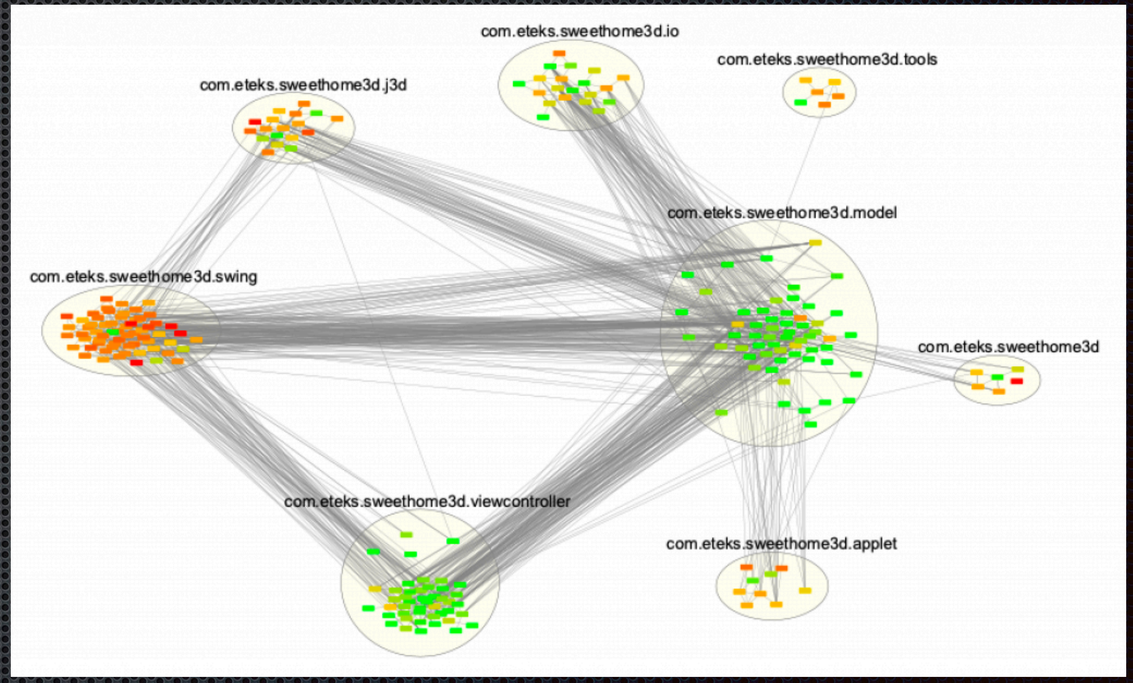
- Problème générale de l’obsolescence logicielle : émulation, polyfills, etc.
- JSweet : une approche de transpilation ouverte avec du mapping de dépendances (substituer une librairie par une autre plus adaptée)
  - BigDecimal => Big.js
- Application à SweetHome3D
  - http://www.sweethome3d.com/



Source : https://renato.athaydes.com/posts/comparing-jvm-alternatives-to-js.html

Aide à la décision pour la modernisation

- Migrateed, un prototype développé avec JSweet et Cytoscape
- Permet de visualiser la complexité et la lourdeur d'un logiciel de manière macro
  - Détection de cycles
  - Clusterisation
  - Propagation
  - Aide à la décision



Visualisation du code source du logiciel Open Source Sweet Home 3D avec MIGRATEED, prototype développé par CINCHEO pour aider à la migration et à l'optimisation des dépendances.

Importance des langages

- Python est 1.3 x plus rapide juste en changeant les options de compilations
  - https://bugs.python.org/issue38980?fbclid=IwAR0cyfahpBywNzbqLCpcfwatOaU6W8UQp7LgVu8KZ8hDc-wKLjLE7wjOs7g
- Python sur GraalVM (LLVM) - 5 à 6 x plus rapide que l'implémentation CPython (après Warm-up) et 6 à 7 x plus rapide que JPython
  - https://www.graalvm.org/python/quickstart
- Optimisation du langage INI développé par CINCHEO (GraalVM / Truffle) : gain x1000 de performance sur du calcul fonctionnel pur
  - https://github.com/cincheo/ini v.s. https://github.com/Saauan/ini-scratch

Transparence et auto-optimisation

- Xiao Y, Nazarian S, Bogdan P. Self-optimizing and self-programming computing systems: A combined compiler, complex networks, and machine learning approach. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2019 Mar 1;27(6):1416-27.
- Doddapaneni K, Ever YK. Evaluation of Simulation Approaches and Need for MDE in Energy Efficiency, Performance and Availability Assessment of IoT. In Performability in Internet of Things 2019 (pp. 33-46). Springer, Cham.
- Georgiou K, Kerrison S, Chamski Z, Eder K. Energy transparency for deeply embedded programs. ACM Transactions on Architecture and Code Optimization (TACO). 2017 Mar 21;14(1):1-26.

Conclusion

Conclusion (1/2)

- L’éco-conception nécessite une démarche méthodologique globale et des outils pour mesurer et maximiser l’efficacité en fonction des spécifications et des usages
  - Cadre pour l’optimisation
- Importance du monitoring (statistiques d’usages et de consommation en resources)
- Définition des contraintes de QoS
  - Par exemple : le serveur de rapports est moins rapide avec 25 utilisateurs, mais il reste dans le cadre des contraintes de QoS fixées (< 1 seconde par rapport)
- L’optimisation du logiciel et la manière de développer le logiciel représente un effet de levier essentiel pour l’éco-conception de services numérique

Conclusion (2/2)

- Il faudra des outils complexes pour bien maitriser les champs des possibles
  - Outils et méthodes d'optimisation et d'aide à la décision multi-critères (algorithmes, empreinte mémoire, utilisation du réseau, maîtrise des dépendances, …)
  - Allocation des ressources et déploiement au bénéfice de l'efficacité énergétique (dépend fortement de l'usage attendu)
  - Développement de langages (et des compilateurs) capables de pendre en compte la notion d'efficacité (transparence et auto-adaptation)
- La recherche a un rôle clé à jouer en partenariat avec les ESN pour accompagner cette transition de manière durable et efficace
  - **Partenariats de R&D**
  - **Partenariats pour la formation**