le cnam

SMB215

2ème semestre 2020/2021

Green Computing

Partie 1

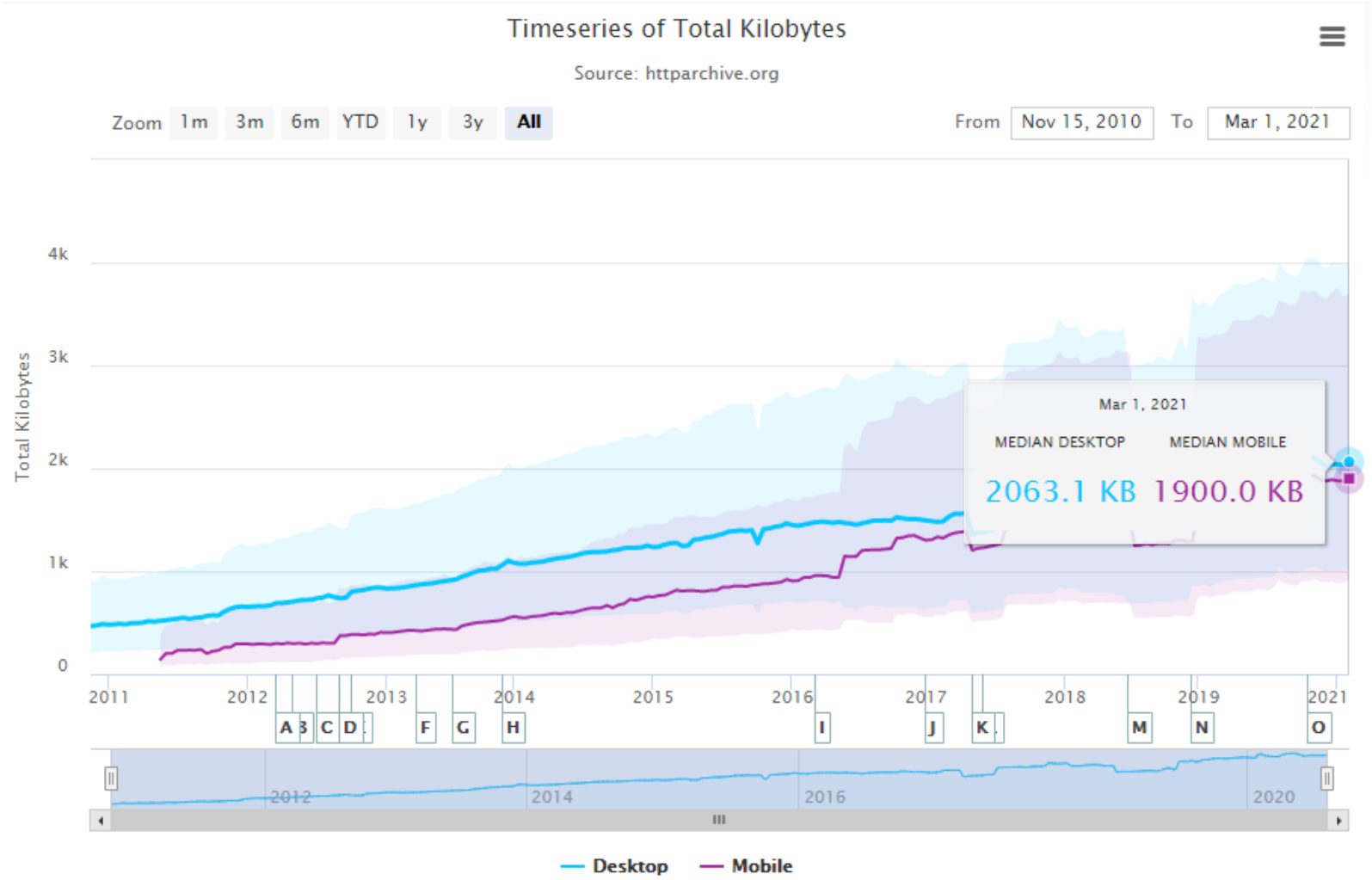
27 Avril 2021

Alexandra SEMEUR

Alexandre TORTEVOIS

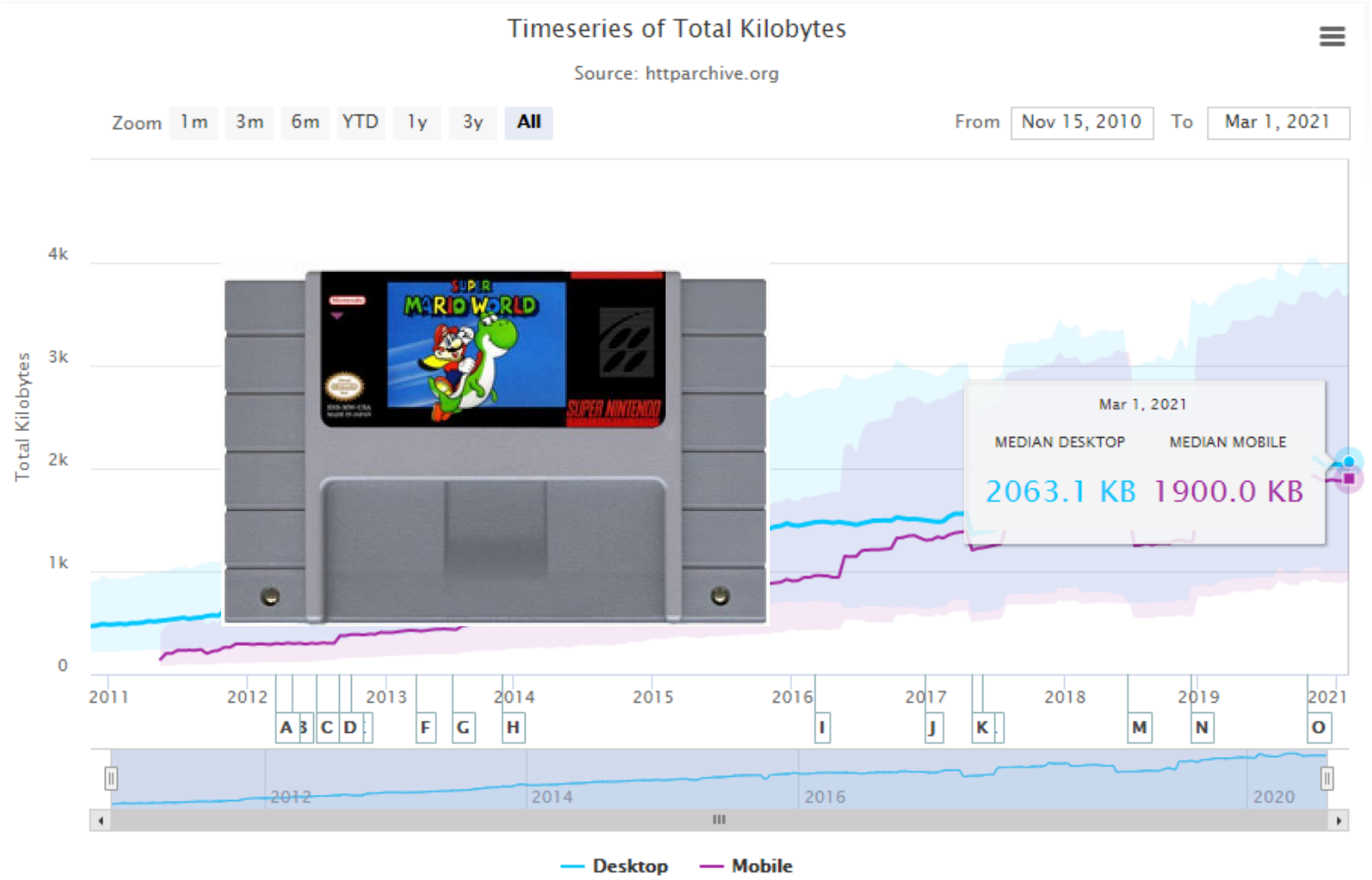
Introduction – Constat

- Poids moyen d'une page Web : 2Mo → 1.76g CO_2 / vue



Introduction – Constat

- Poids moyen d'une page Web : 2Mo → 1.76g CO_2 / vue



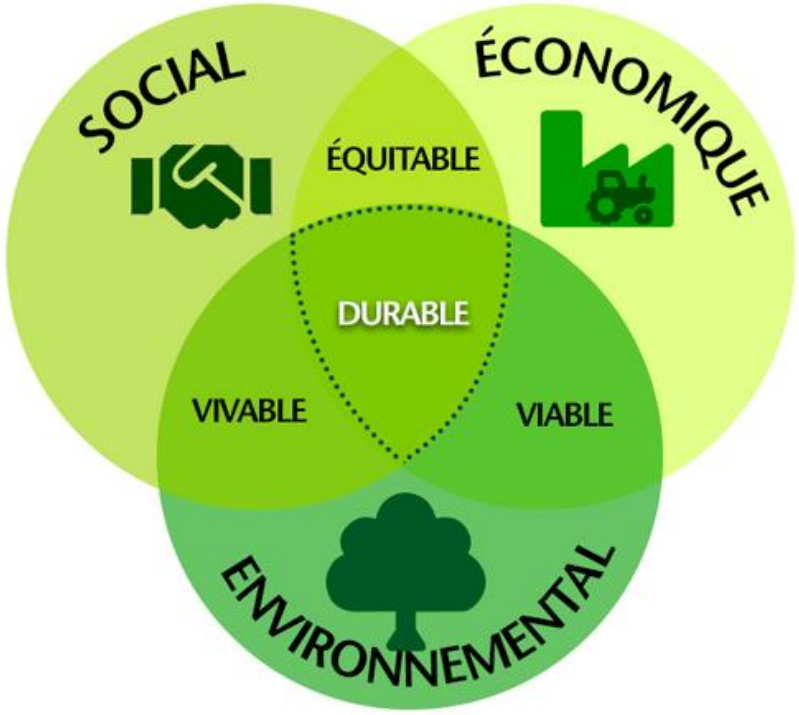
Constat

- Les industries du numérique représentent
  - 10% de la consommation totale d'électricité dans le monde
  - **plus** de 2 % de toutes les émissions de CO_2

- Le secteur des activités numériques représente entre 5 et 10 % de l'impact environnemental de la France

- Datacenters :14 % de l'empreinte carbone du numérique français.
  - **o**bjectifs : réduction de la consommation énergétique
    - de 40 % d'ici 2030
    - de 60% d'ici 2050

Enjeux



Réduire l'empreinte écologique, économique et social, tout au long du cycle de vie d'un produit, de sa conception à son recyclage

« Synonymes » de :

- **Développement durable**
- **Écoconception**
- **Transition écologique**

Notre approche

1. Histoire du Green Computing
   - **Normes & écolabels**
   - **Législation**

2. **Moyens et techniques**
   - **Gestion de ressources**
   - **Matériel**
   - **Virtualisation**
   - **Monitoring**

3. Expérimentation

Gestion des ressources

Production d'énergie verte :

- Eolien
- Solaire
- Géothermie

Réduction consommation d'eau :

- Optimisation refroidissement
- Réduction consommation lors du processus de production

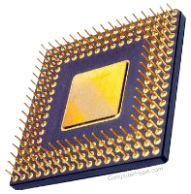
Sensibilisation aux écogestes, écoconception :

- **Arrêt VS Veille**
- **WiFi/Filaire VS 4G/5G**
- **Limite / Allégement flux**

Matériel

SSD au lieu HDD

Stockage



Finesse de gravure

Processeur

Adapter à l'utilisation

Carte graphique

LCD et limiter la taille

Moniteur

Alimentation électrique intelligente

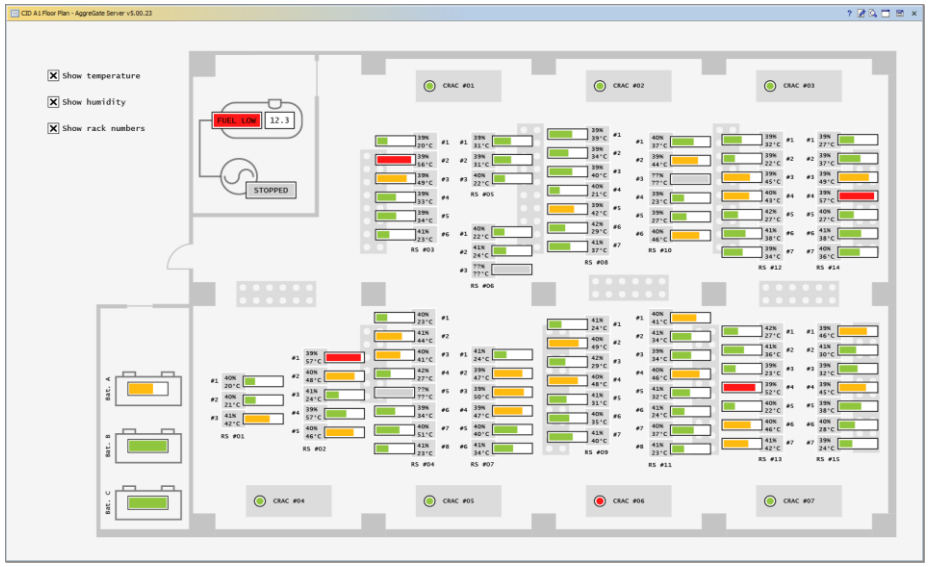
Alimentation

Virtualisation & Monitoring

Virtualisation : Réduction du nombre de serveurs physiques

Monitoring : permet d'adapter les ressources aux besoins

- **Sondes d'environnement (température, hygrométrie)**
- **Capteurs de consommation électrique**

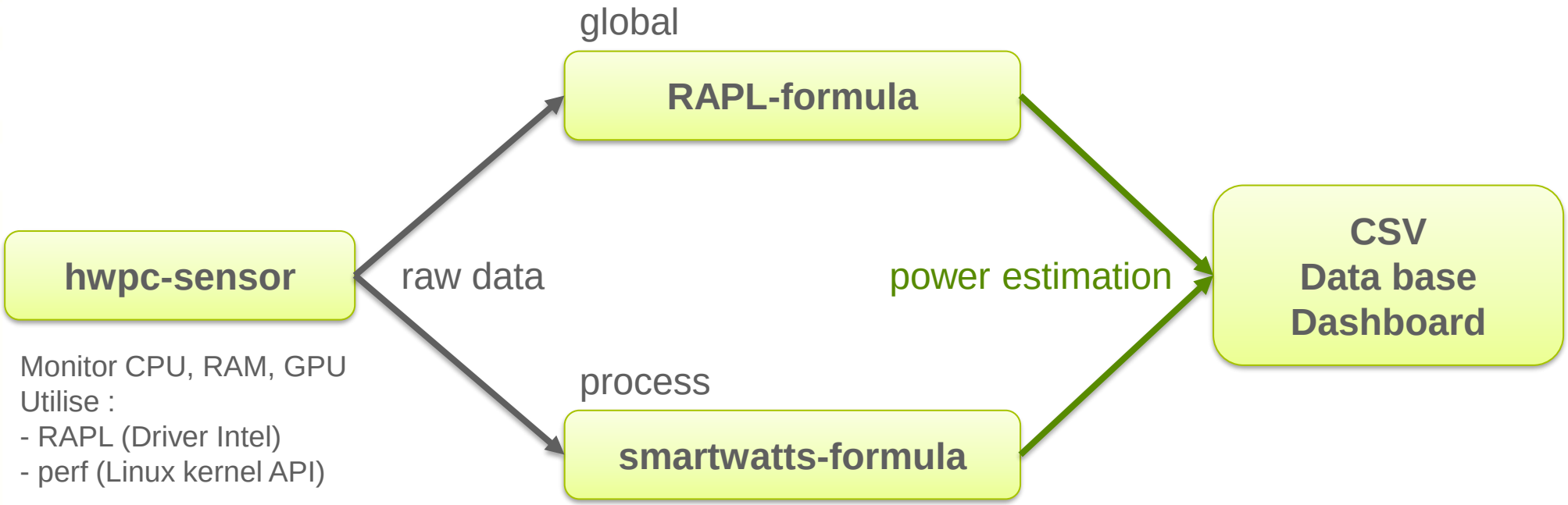


Solution de monitoring

PowerAPI

- Projet développé par INRIA et Université de Lille depuis 2013
- Librairie pour estimer l'énergie consommée par un processus en temps réel
- https://github.com/powerapi-ng
- version 0.9.2 du 10 Mars 2021

Solution de monitoring



Monitor CPU, RAM, GPU
Utilise :
- RAPL (Driver Intel)
- perf (Linux kernel API)

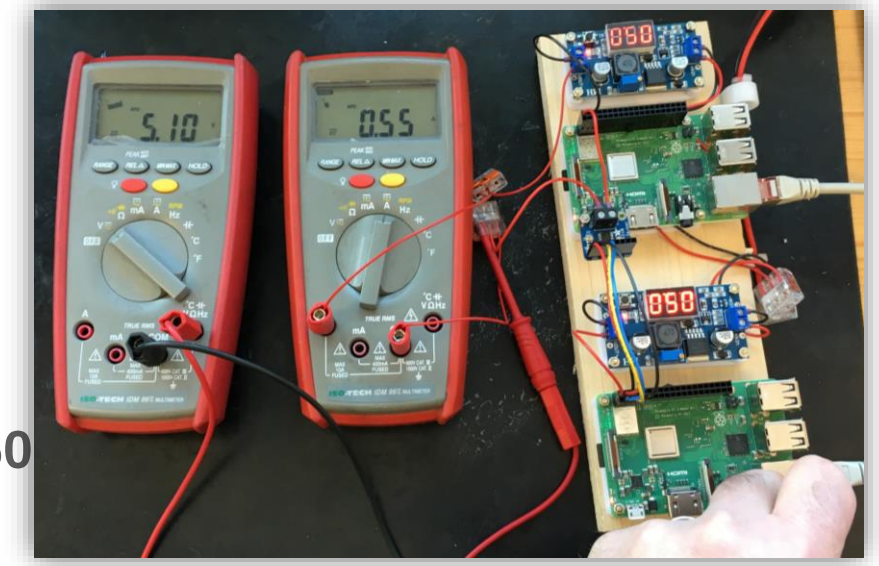
Limitations :
- Processeur Intel
- Incompatible avec une VM
- Orienté « docker »

Expérimentation

- Mesurer et comparer l'efficience de quelques langages de programmation :
  - **Coût CPU**
  - **Temps d'exécution**
  - **Impact mémoire**
  - **Consommation énergétique**

- Utilisation d'outils intégré à Linux :
  - **perf : analyse CPU**
  - **time : analyse Mémoire**

- Matériels :
  - **Raspberry PI 3B+**
  - **Capteur de courant / tension TI INA260**



- Comparaison résultats avec littérature

QUESTIONS

MERCI DE VOTRE ATTENTION

le cnam

SMB215

2ème semestre 2020/2021

Green Computing

Présentation du 8 juin 2021

Alexandra SEMEUR

Alexandre TORTEVOIS

Résumé de la présentation du 6 avril

- Comment rendre l'informatique "durable" ?
  - Gestion des ressources
    - Utilisation d'énergie verte
    - Réduction de la consommation d'eau
    - Sensibilisation aux écogestes et à l'éco-conception
  - Matériel
  - Virtualisation

- Monitoring

  Focus sur un outils en particulier : PowerAPI

Sommaire

1. Ecolabels / Normes / Aspects Législatif
2. Expérimentation
3. Conclusion
4. Lien avec les autres exposés

Ecolabels

Premier écolabel créé en Allemagne en 1977 : Blue Angel

Plus de **30 normes à travers le monde** :

- 3 normes privées créées par des entreprises pour leurs propres produits
- Labels principalement par pays
- Évaluation plus ou moins indépendante de l'impact du cycle de vie et/ou de l'efficacité énergétique

Le plus utilisé : Energy Star

Une organisation, le **GEN (Global Ecolabelling Network)** pour fédérer les écolabels autour de critères communs

Normes

ISO : International Organization for Standardization

- Fondée en 1947
- Indépendante

ISO 26000 — Responsabilité sociétale
2010

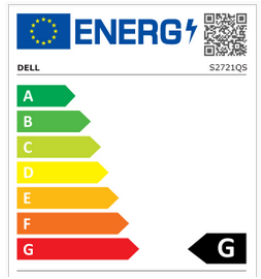
ISO 14001 — Système de Management Environnemental
2015

ISO 50001 — Système de Management de l'Énergie
2018

Aspects Législatif – Europe

- 2003 : Directive RoHS
  - **Limiter l'emploi de substances dangereuses**

- 2019 : Label 2020
  - **Nouvelle étiquette énergétique**
  - **Concerne les écrans d'ordinateurs**



- 2019 : Green Deal
  - **Être climatiquement neutre à horizon 2050**

Aspects Législatif – France

- 2020 : Loi Grenelle 2
  - Impose l'établissement d'un bilan des émissions de gaz à effet de serre

- 2020 : Loi Anti-gaspillage pour une économie circulaire
  - **Article 16 introduit indice de réparabilité**
  - **Concerne smartphones, ordinateurs portables**

- 2021 : Loi Climat & Résilience
  - **Former à l'écoconception logicielle**

Expérimentation

- Étudier l'impact du langage de programmation sur la consommation énergétique

- Étude Université de Minho (Portugal) en 2017
  - 27 langages
  - 10 algorithmes

Time & Memory	Energy & Time	Energy & Memory	Energy & Time & Memory
C • Pascal • Go	C	C • Pascal	C • Pascal • Go
Rust • C++ • Fortran	Rust	Rust • C++ • Fortran • Go	Rust • C++ • Fortran
Ada	C++	Ada	Ada
Java • Chapel • Lisp • Ocaml	Ada	Java • Chapel • Lisp	Java • Chapel • Lisp • Ocaml
Haskell • C#	Java	OCaml • Swift • Haskell	Swift • Haskell • C#
Swift • PHP	Pascal • Chapel	C# • PHP	Dart • F# • Racket • Hack • PHP
F# • Racket • Hack • Python	Lisp • Ocaml • Go	Dart • F# • Racket • Hack • Python	JavaScript • Ruby • Python
JavaScript • Ruby	Fortran • Haskell • C#	JavaScript • Ruby	TypeScript • Erlang
Dart • TypeScript • Erlang	Swift	TypeScript	Lua • JRuby • Perl
JRuby • Perl	Dart • F#	Erlang • Lua • Perl	
Lua	JavaScript	JRuby	
	Racket		
	TypeScript • Hack		
	PHP		
	Erlang		
	Lua • JRuby		
	Ruby		

Comment monitorer une exécution ?

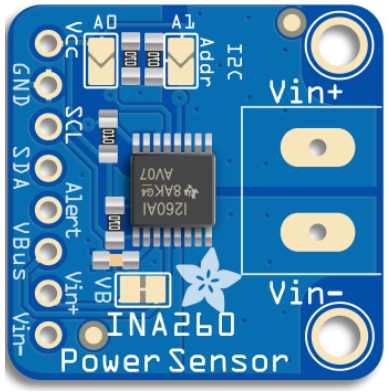
- Consommation électrique :

$$E_{nergie} = P_{uissance} \times T_{emps}$$

- Mesurer la puissance électrique : wattmètre

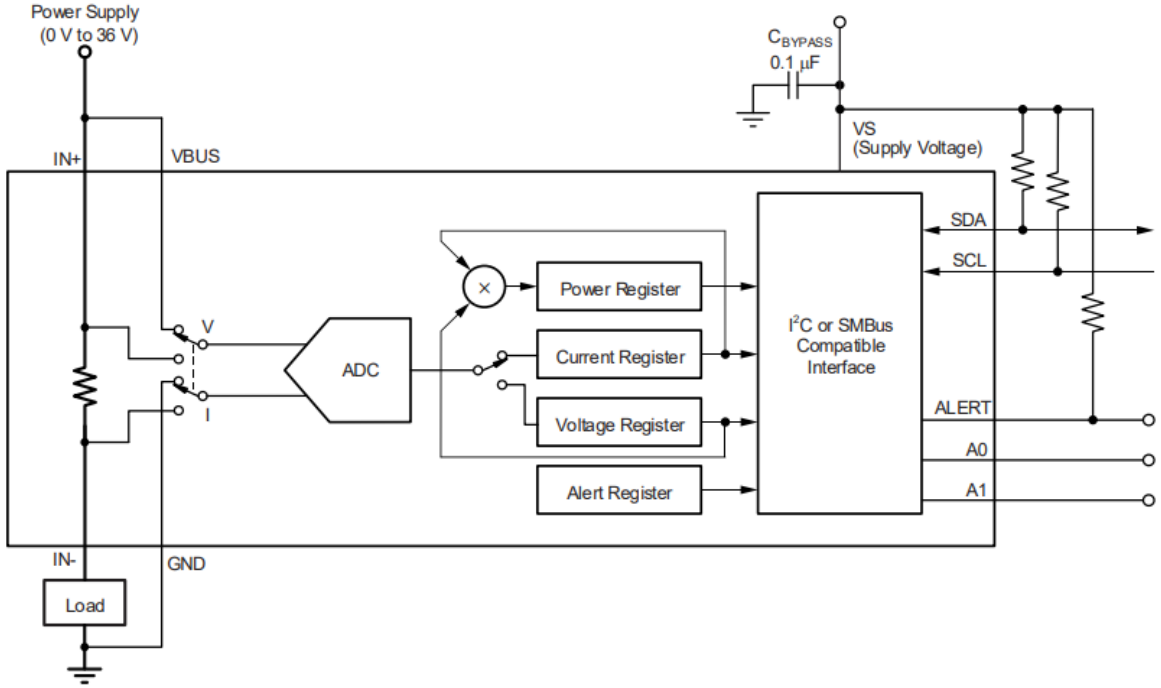
- Implémentation :
  - Capteur Texas Instrument - INA260
  - Commandes Linux
    - `perf` : stats hardware (sous réserve d'avoir un CPU compatible)
    - `time` : stats usage mémoire

Capteur Texas Instrument - INA260



- Interface I²C / SMBus, drivers inclus dans Linux
- Précision : Erreur 0,15% (maximum)
- Programmable : Moyenne, Temps d'acquisition
- Registres internes :
  - Courant
  - Tension

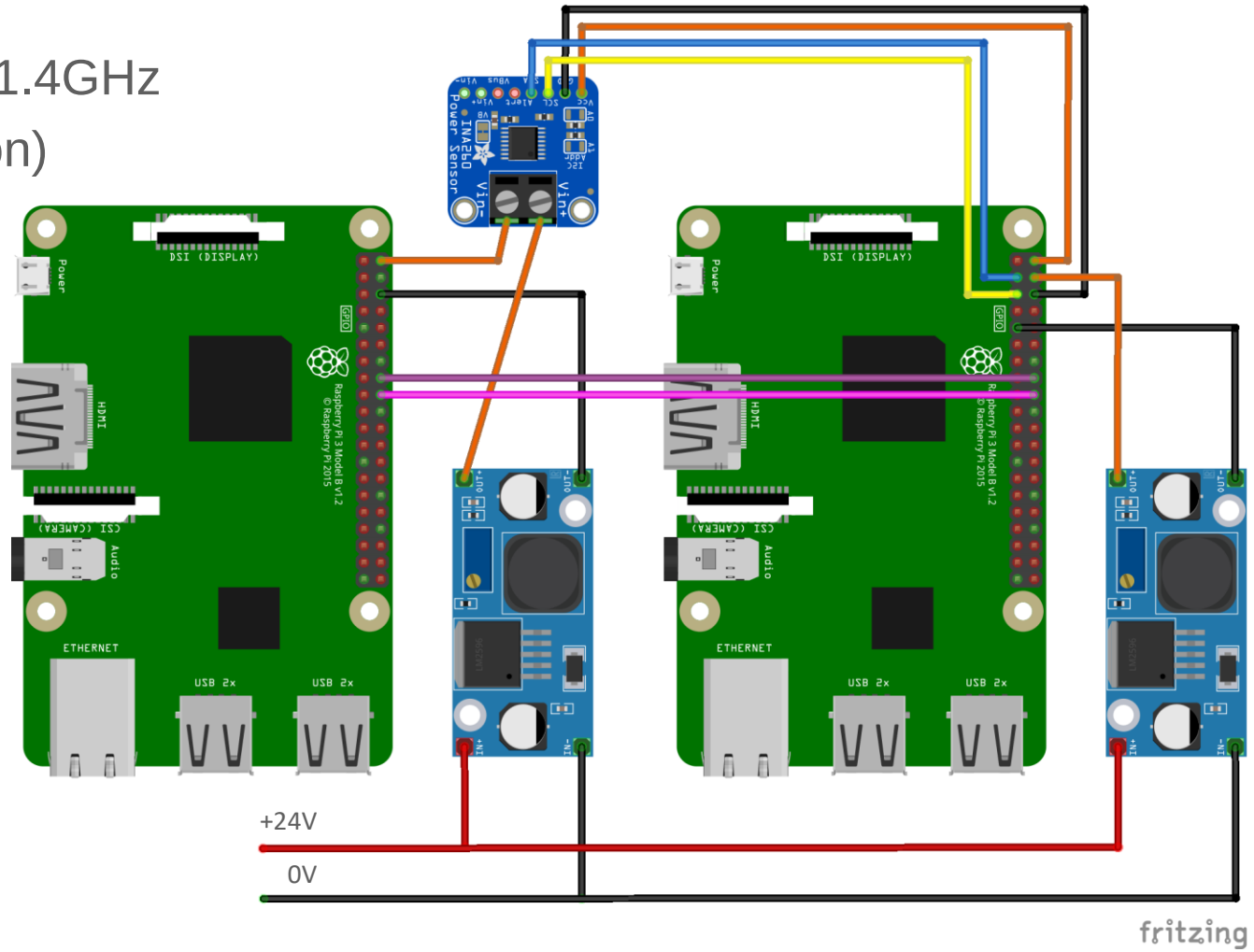
https://www.ti.com/product/INA260



Intégration : Banc de tests

Raspberry Pi 3B+ :

- SoC (ARMv8) 64-bit @ 1.4GHz
  - 16KB L1P (Instruction)
  - 16KB L1D (Data)
  - 512KB L2
- 1GB LPDDR2 SDRAM

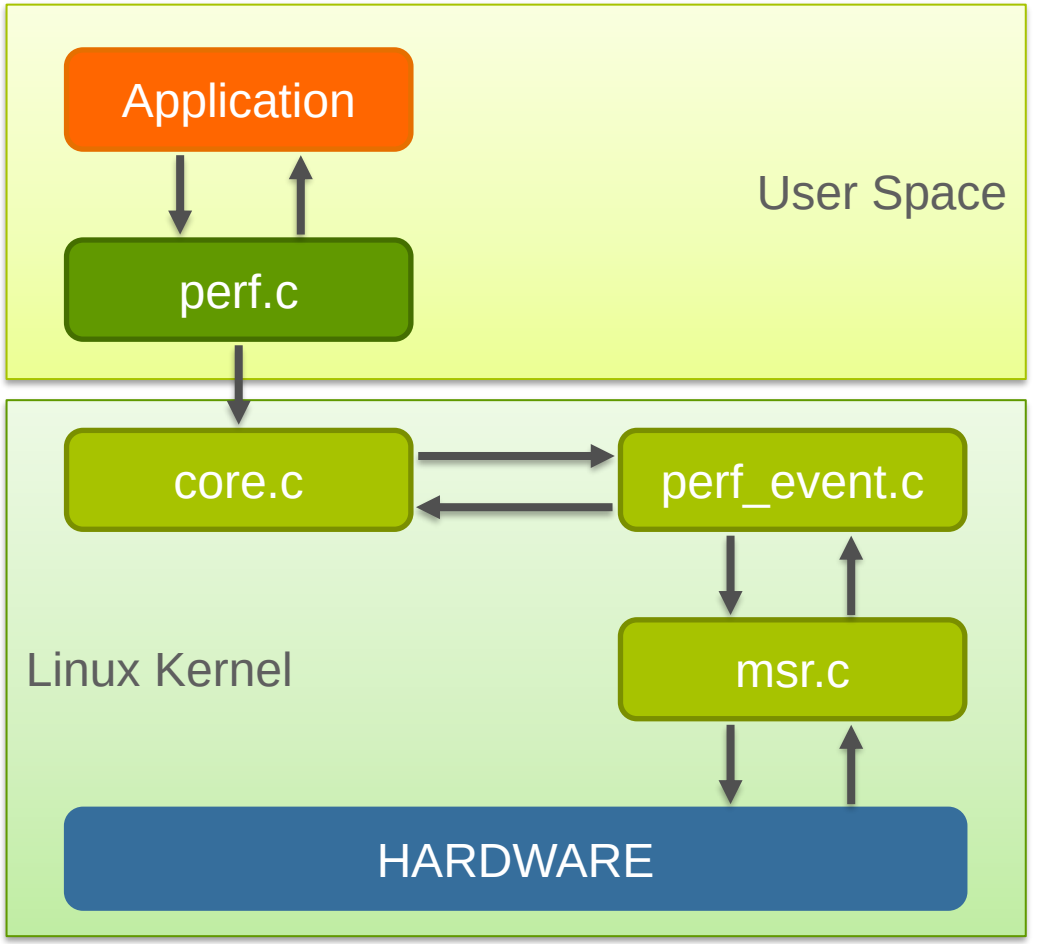


RPi « Test »

RPi « Centrale Acquisition »

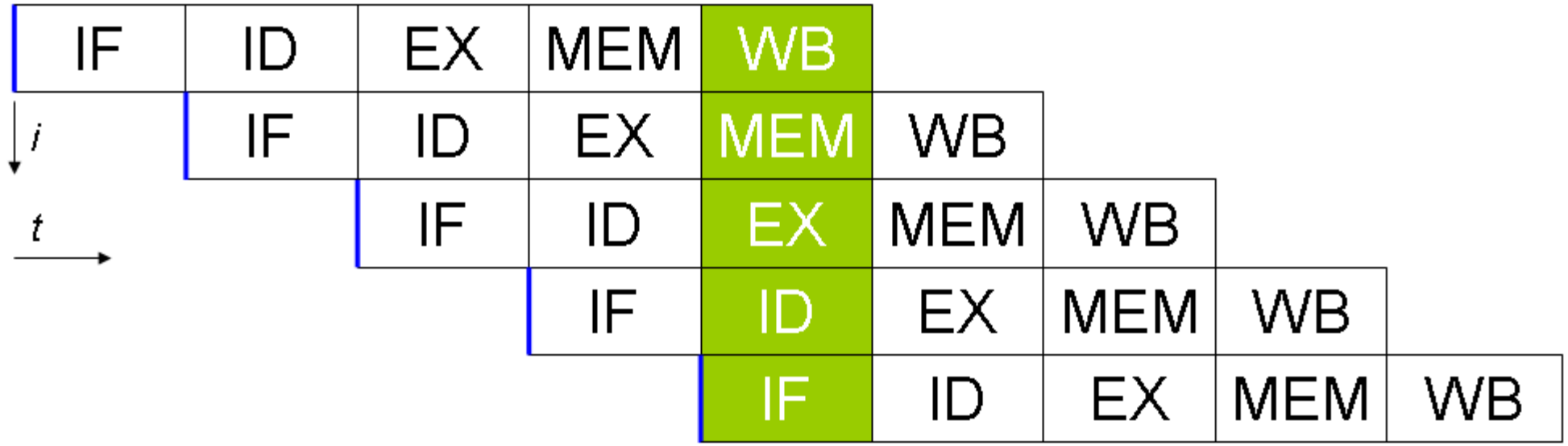
Commande Linux `perf`

- Profileur officiel Linux (natif)
- Surveillance des performances
  matérielles et logicielles
- Faible surcharge par rapport au profilage basé sur l'instrumentation



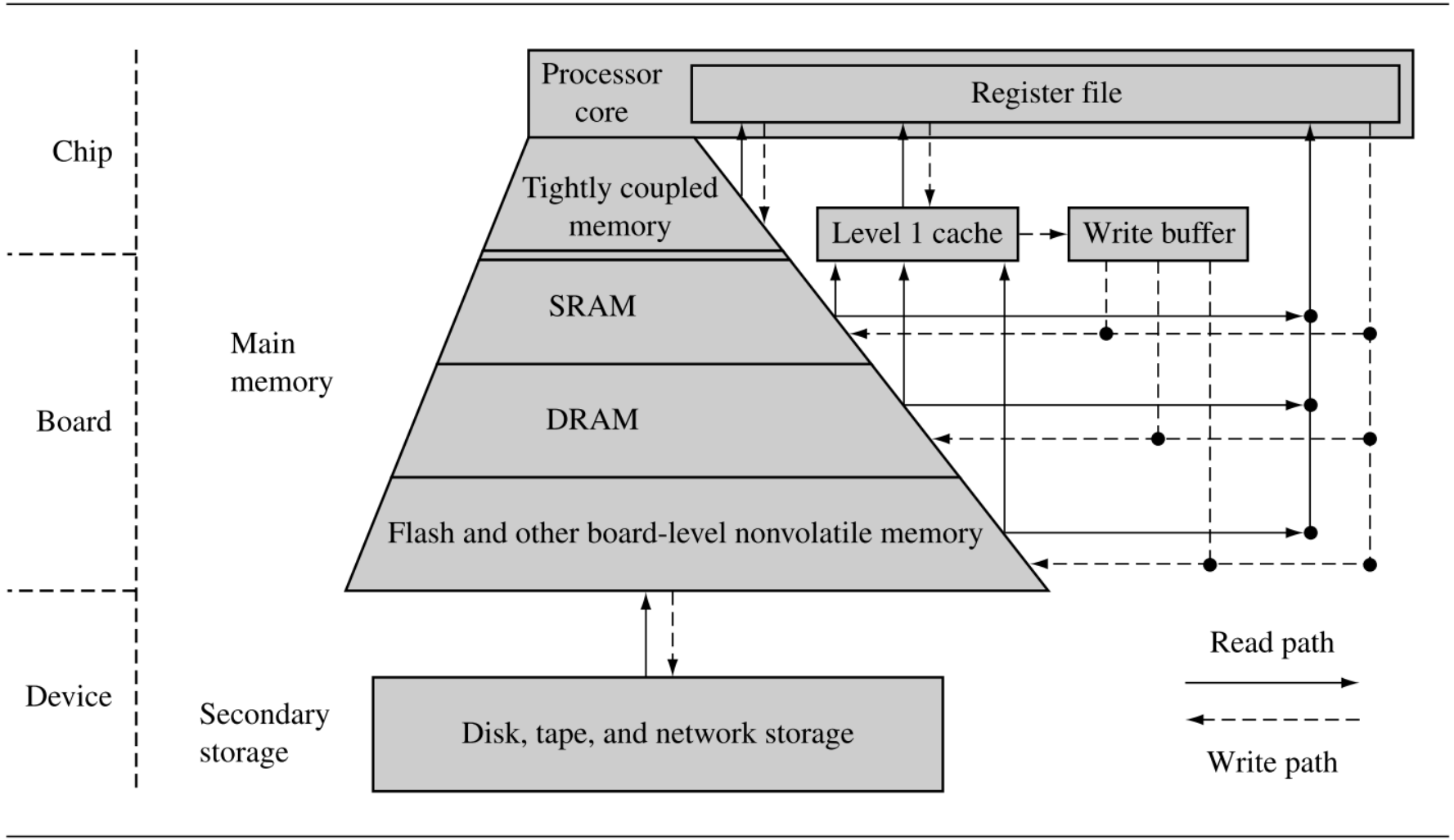
https://github.com/torvalds/linux/tree/master/tools/perf

Pipeline



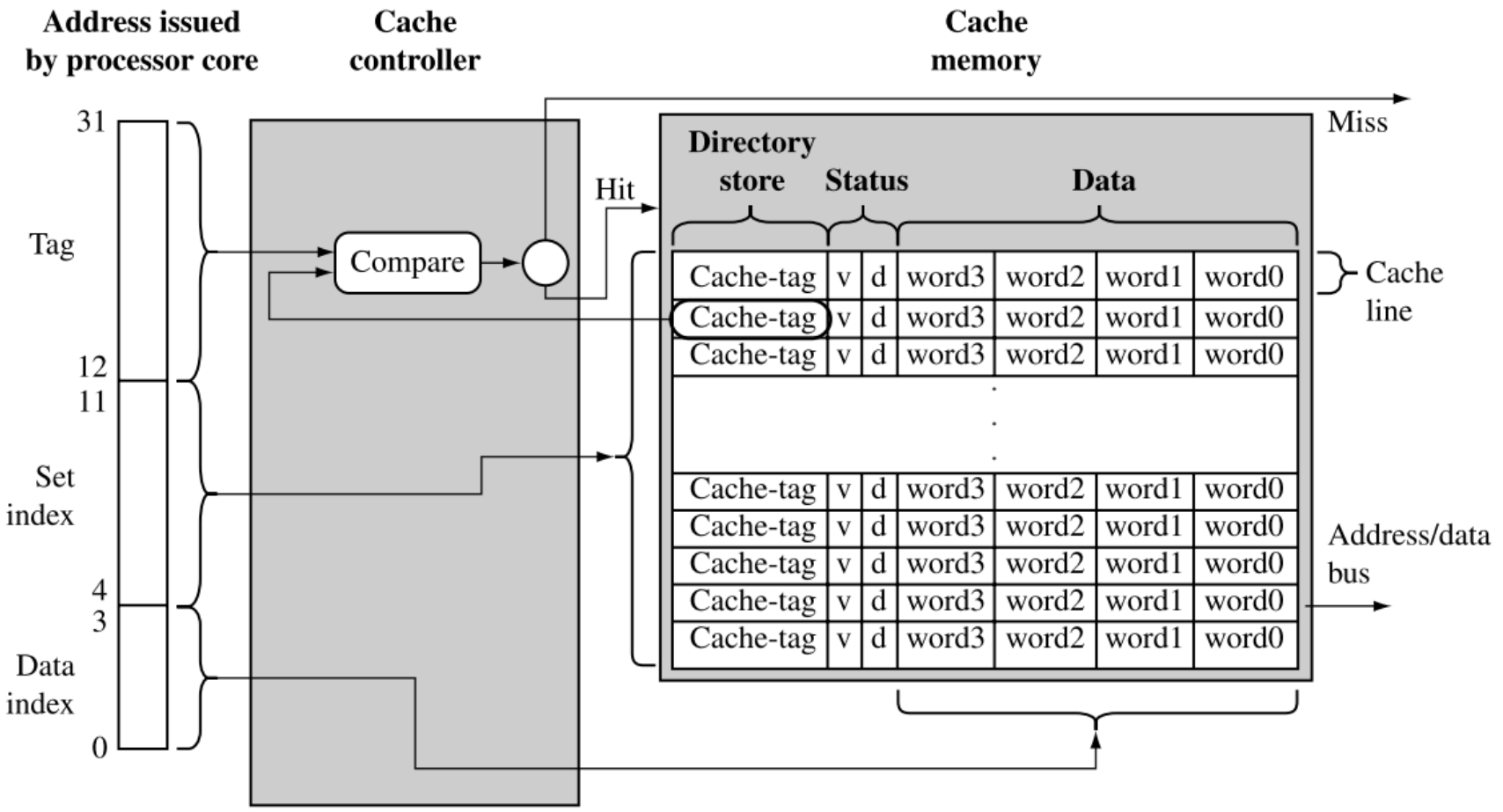
Source : Wikipédia

Hiérarchie mémoire



Source : ARM System Developer's Guide Designing and Optimizing System Software p.404

Architecture Cache



Source : ARM System Developer's Guide Designing and Optimizing System Software p.404

$: taskset -c 0 /usr/bin/perf stat -d -- ./algo-03/art-optimal-linkedlist.o

```
 Performance counter stats for './algo-03/art-optimal-linkedlist.o':

      16611.525986      task-clock (msec)         #    0.999 CPUs utilized
                69      context-switches          #    0.004 K/sec
                 0      cpu-migrations            #    0.000 K/sec
               876      page-faults               #    0.053 K/sec
       23191635431      cycles                    #    1.396 GHz                      (87.48%)
       20689794366      instructions              #    0.89  insn per cycle           (87.53%)
        2434210024      branches                  #  146.537 M/sec                    (87.48%)
           7637792      branch-misses             #    0.31% of all branches          (87.48%)
       10062646536      L1-dcache-loads           #  605.763 M/sec                    (87.46%)
           5033635      L1-dcache-load-misses     #    0.05% of all L1-dcache hits    (87.54%)
         267545180      LLC-loads                 #   16.106 M/sec                    (87.54%)
         133078484      LLC-load-misses           #   49.74% of all LL-cache hits     (74.96%)

      16.636415260 seconds time elapsed
```

```
$: taskset -c 0 /usr/bin/time -v ./algo-03/art-optimal-linkedlist.o
        Command being timed: "./algo-03/art-optimal-linkedlist.o"
        User time (seconds): 16.61
        System time (seconds): 0.01
        Percent of CPU this job got: 99%
        Elapsed (wall clock) time (h:mm:ss or m:ss): 0:16.65
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 4056
        Average resident set size (kbytes): 0
        Major (requiring I/O) page faults: 0
        Minor (reclaiming a frame) page faults: 907
        Voluntary context switches: 5
        Involuntary context switches: 86
        Swaps: 0
        File system inputs: 0
        File system outputs: 96
        Socket messages sent: 0
        Socket messages received: 0
        Signals delivered: 0
        Page size (bytes): 4096
        Exit status: 0
```

Le problème

Version simplifiée d'un problème posé lors du Google Hash Code 2014

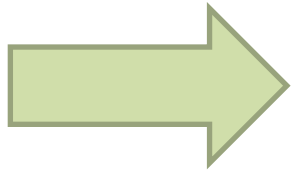
L'objectif est de « peindre » une image (noir/blanc) en un minimum d'opérations.

Deux opérations sont possibles :

- **FILL,x,y,size** : peint un carré de taille *size* dont le côté en haut et à gauche est en *x, y*
- **ERASE,x,y** : efface la peinture du pixel en *x, y*

Exemple d'entrée :

6,5
###**#
###***
###***
***#**
***##*



Exemple de sortie :

Le problème

Version simplifiée d'un problème posé lors du Google Hash Code 2014

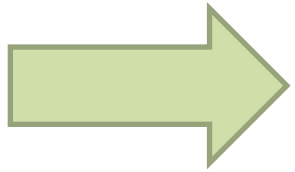
L'objectif est de « peindre » une image (noir/blanc) en un minimum d'opérations.

Deux opérations sont possibles :

- **FILL,x,y,size** : peint un carré de taille *size* dont le côté en haut et à gauche est en *x, y*
- **ERASE,x,y** : efface la peinture du pixel en *x, y*

Exemple d'entrée :

6,5
###**#
###***
###***
***#**
***##*



Exemple de sortie :

Le problème

Version simplifiée d'un problème posé lors du Google Hash Code 2014

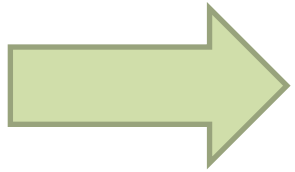
L'objectif est de « peindre » une image (noir/blanc) en un minimum d'opérations.

Deux opérations sont possibles :

- **FILL,x,y,size** : peint un carré de taille *size* dont le côté en haut et à gauche est en *x, y*
- **ERASE,x,y** : efface la peinture du pixel en *x, y*

Exemple d'entrée :

6,5
###**#
#######
#######
***#**
******



Exemple de sortie :

FILL,0,0,3

Le problème

Version simplifiée d'un problème posé lors du Google Hash Code 2014

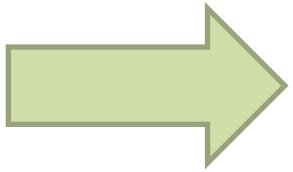
L'objectif est de « peindre » une image (noir/blanc) en un minimum d'opérations.

Deux opérations sont possibles :

- **FILL,x,y,size** : peint un carré de taille *size* dont le côté en haut et à gauche est en *x, y*
- **ERASE,x,y** : efface la peinture du pixel en *x, y*

Exemple d'entrée :

6,5
###**#
###***
###***
***#**
***##*



Exemple de sortie :

FILL,0,0,3
FILL,5,0,1

Le problème

Version simplifiée d'un problème posé lors du Google Hash Code 2014

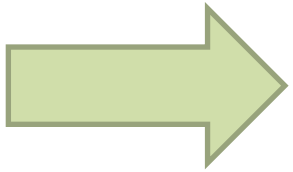
L'objectif est de « peindre » une image (noir/blanc) en un minimum d'opérations.

Deux opérations sont possibles :

- **FILL,x,y,size** : peint un carré de taille *size* dont le côté en haut et à gauche est en *x, y*
- **ERASE,x,y** : efface la peinture du pixel en *x, y*

Exemple d'entrée :

6,5
###**#
###***
###***
***#**
***###*



Exemple de sortie :

FILL,0,0,3
FILL,5,0,1
FILL,3,3,2

Le problème

Version simplifiée d'un problème posé lors du Google Hash Code 2014

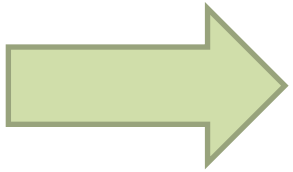
L'objectif est de « peindre » une image (noir/blanc) en un minimum d'opérations.

Deux opérations sont possibles :

- **FILL,x,y,size** : peint un carré de taille *size* dont le côté en haut et à gauche est en *x, y*
- **ERASE,x,y** : efface la peinture du pixel en *x, y*

Exemple d'entrée :

6,5
###**#
###***
###***
***#**
***##*



Exemple de sortie :

FILL,0,0,3
FILL,5,0,1
FILL,3,3,2
ERASE,4,3

Le problème

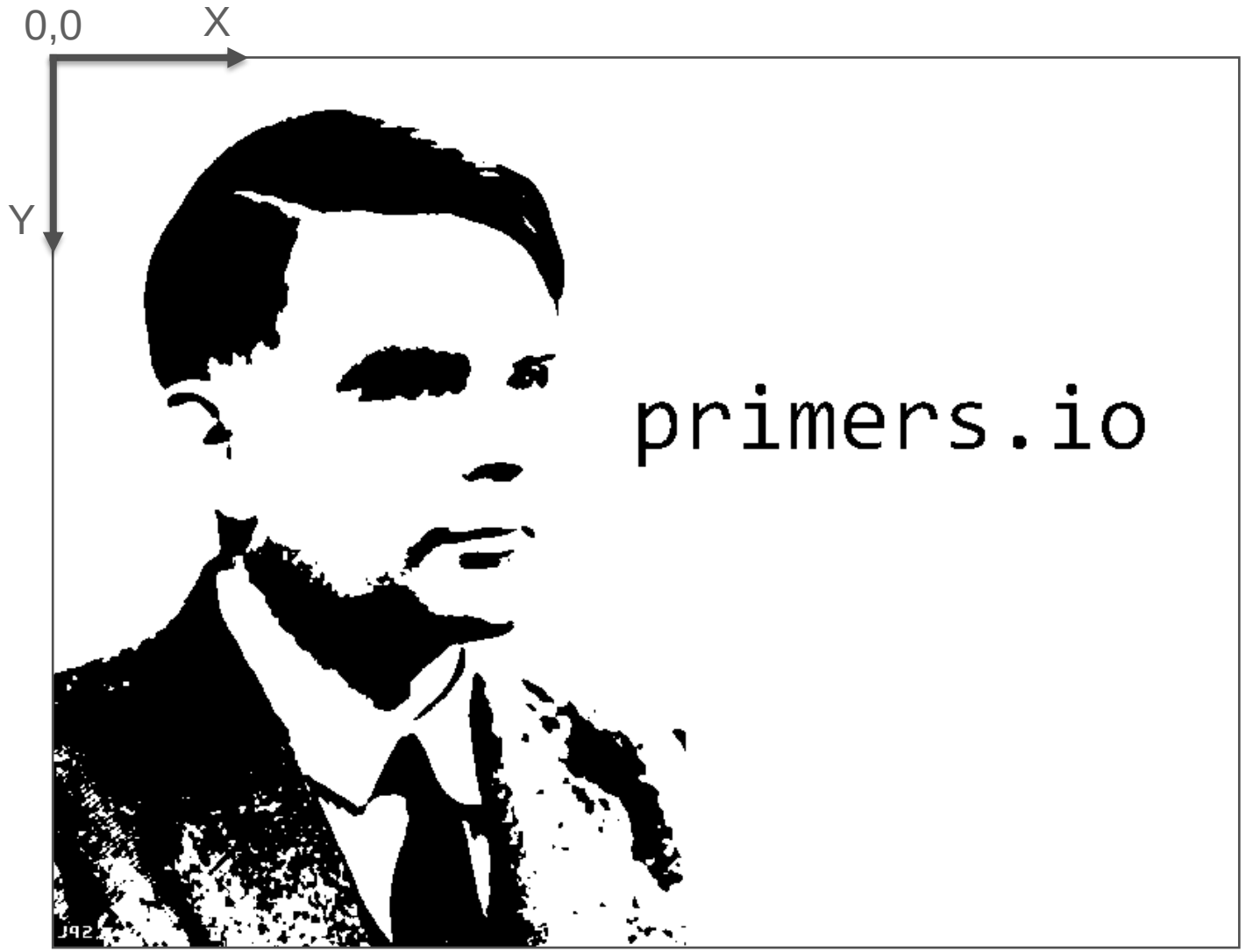


Image : 800 x 600 => 480 000 pixels => 84 784 pixels noir

Algorithme

- Solution n°1 « naïve » :
  - Peindre les pixels 1 par 1

  84 784 lignes en sortie

Algorithme

- Solution n°1 « naïve » :
  - Peindre les pixels 1 par 1

  84 784 lignes en sortie

- Solution n°2 :
  - Pré-calcul des « plus grands carrés » possibles pour chaque pixel
  - Parcourir l'image et remplir par le carré le plus efficient

  3 146 lignes en sortie

Algorithme

- Solution n°1 « naïve » :
  - Peindre les pixels 1 par 1

  84 784 lignes en sortie

- Solution n°2 :
  - Pré-calcul des « plus grands carrés » possibles pour chaque pixel
  - Parcourir l'image et remplir par le carré le plus efficient

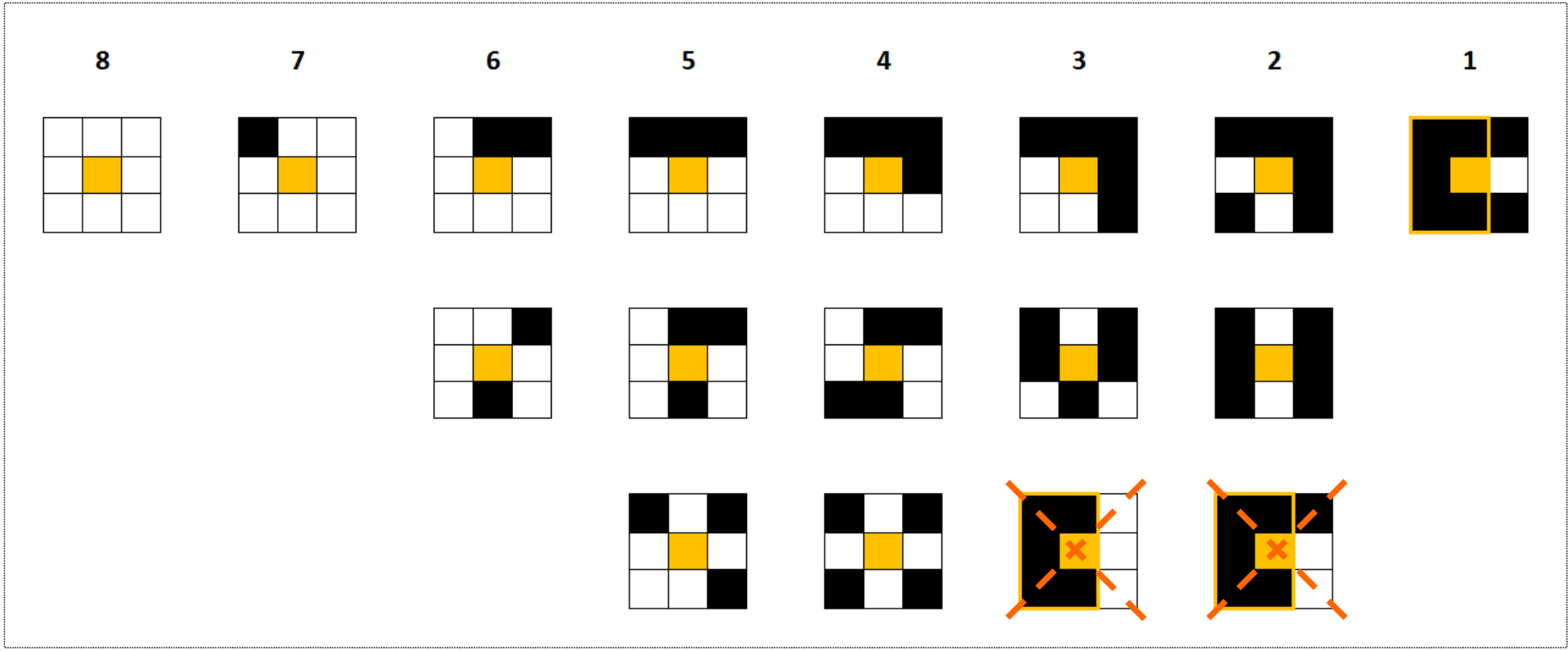
  3 146 lignes en sortie

- Solution n°3 :
  - Pré-calcul des « plus grands carrés » possibles pour chaque pixel
  - Pré-calcul des indices d'isolement de chaque pixel
  - Remplir en priorité les pixels les plus isolés par le carré le plus efficient

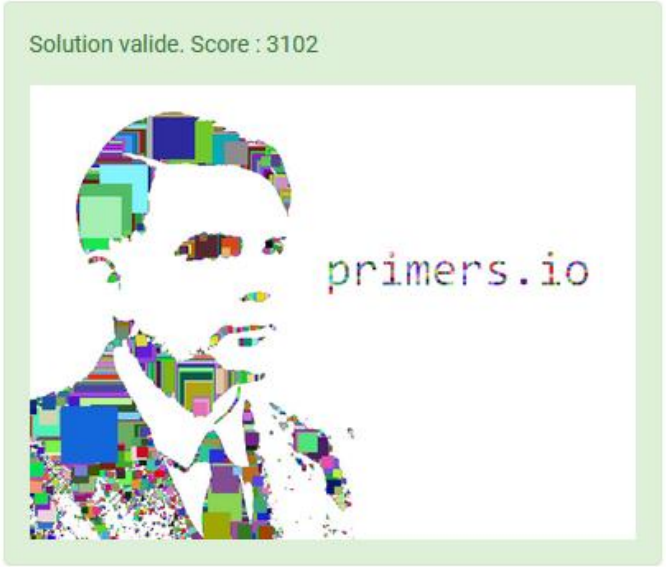
  3 102 lignes en sortie

Algorithme

- Indice d'isolement : nombre de pixels vides autour de la position *x, y*



Algorithme



- Solution n°1 « naïve » :
  - Peindre les pixels 1 par 1

  84 784 lignes en sortie

- Solution n°2 :
  - Pré-calcul des « plus grands carrés » possibles pour chaque pixel
  - Parcourir l'image et remplir par le carré le plus efficient

  3 146 lignes en sortie

- Solution n°3 :
  - Pré-calcul des « plus grands carrés » possibles pour chaque pixel
  - Pré-calcul des indices d'isolement de chaque pixel
  - Remplir en priorité les pixels les plus isolés par le carré le plus efficient

  3 102 lignes en sortie

Implémentation

- 5 langages :


C (référentiel)


Python


Java

PHP

JavaScript

- Exécution bloquée sur un seul cœur : `` `taskset -c 0` ``
- Disclamer : meilleure(s) implémentation(s) ?


https://github.com/a-tortevois/projet-smb215-green-computing

Implémentation (Python)

```python
def getDrawableArea():
    drawableArea = {}
    for y in range(baseY):
        for x in range(baseX):
            if baseGrid[y][x] == "#":
                d = 1
                while True:
                    if not inDrawableArea(x, y, d) or not isDrawable(x, y, d):
                        break
                    d += 1
                drawableArea[(x, y, d)] = d ** 2
                isolationIndex[y][x] = getIsolationIndex(x, y)
    return drawableArea
```

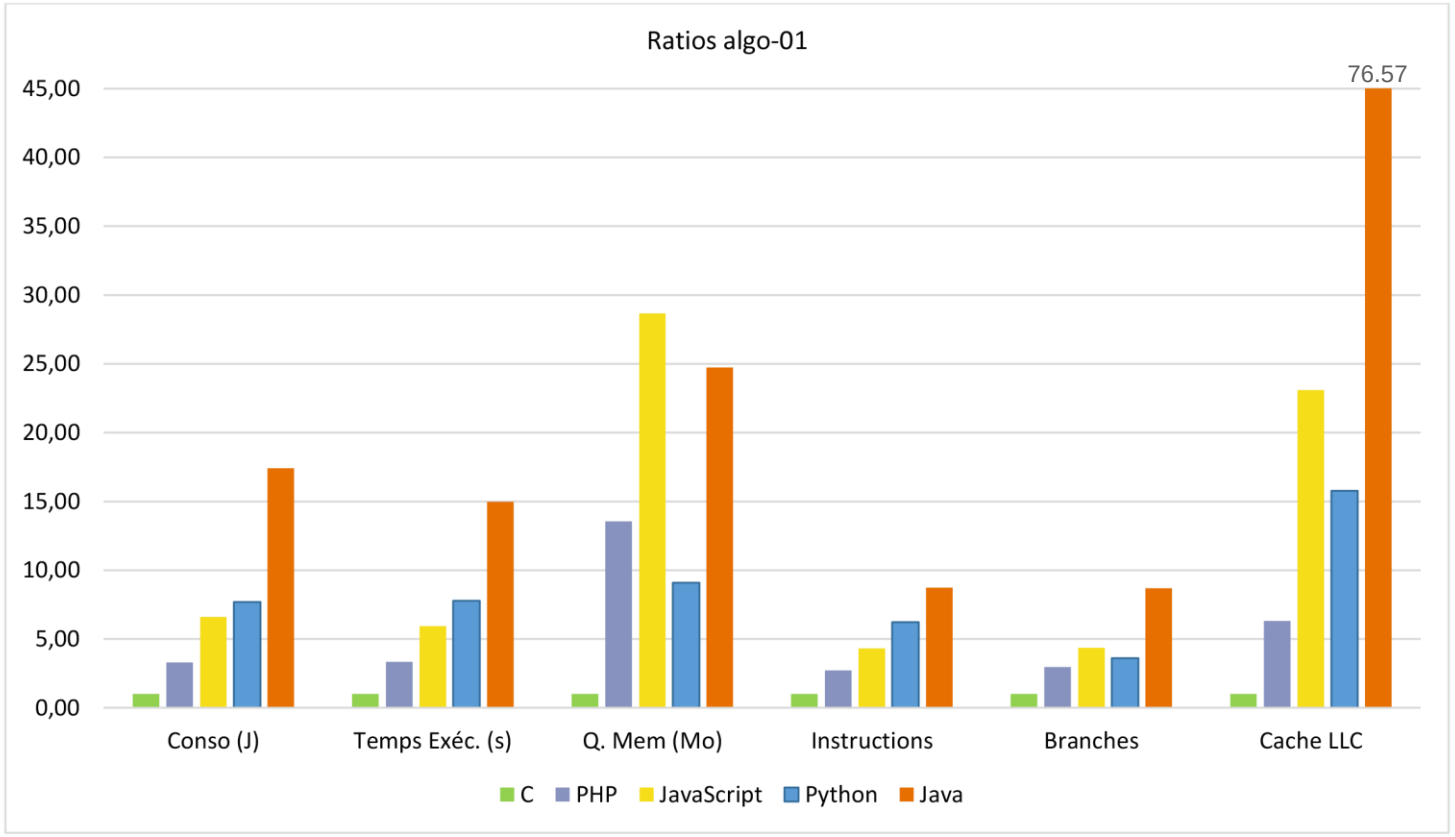
```python
def fillHeaviestArea(x, y):
    found = {}
    listElem = list(drawableArea)
    for elem in listElem:
        if isInArea(x, y, elem):
            weight = computeWeightArea(*elem)
            if weight == 0:
                del drawableArea[elem]
            else:
                found[elem] = weight
    if len(found) > 0:
        elem = list(sortByValueReversed(found))[0]
        addToResult(*elem)
        del drawableArea[elem]
```

Implémentation (Python)

```
with open(INPUT_FILE) as f:
    baseX, baseY = map(int, f.readline().strip().split(','))
    for line in f:
        pixels = list(line.strip())
        baseGrid.append(pixels)
        isolationIndex.append([0] * baseX)

drawableArea = getDrawableArea()

# Fill in the most isolated pixels by heaviest area
for index in range(8, 1, -1):
    if index > 1:
        for y in range(baseY):
            for x in range(baseX):
                if baseGrid[y][x] == "#" and index == int(isolationIndex[y][x]):
                    fillHeaviestArea(x, y)

# Fill in the remaining pixels from the end line
for y in range(baseY - 1, -1, -1):
    for x in range(baseX):
        if baseGrid[y][x] == "#":
            fillHeaviestArea(x, y)

with open(OUTPUT_FILE, 'w') as f:
    f.write('\n'.join(result))
```

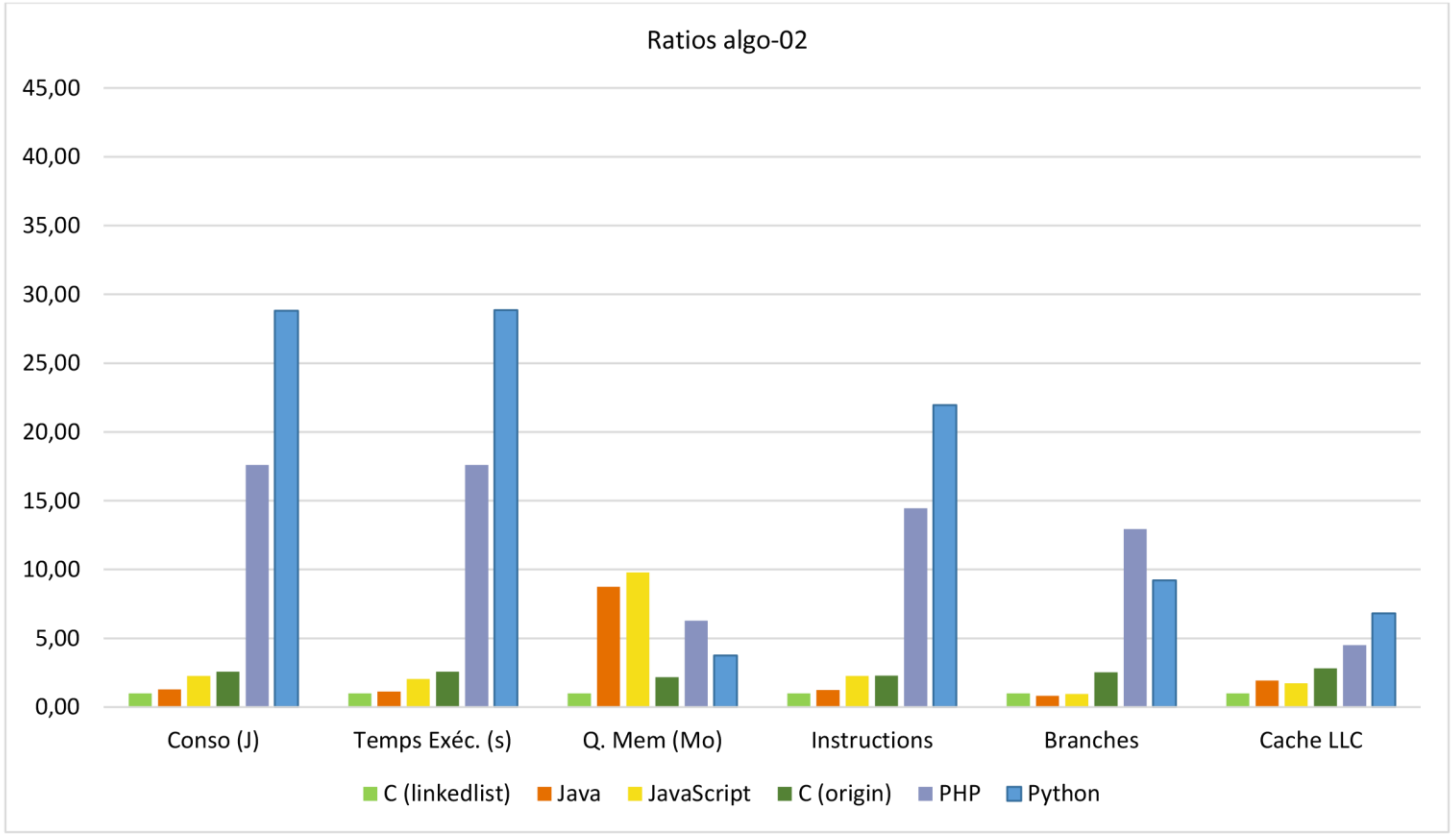
Résultats de l'exécution « Solution n°1 »

Lang.	Conso		Temps Exéc.		Q. Mémoire		Instructions		Branches		Accès Cache LLC	
	(J)	Ratio	(s)	Ratio	(Mo)	Ratio	Ratio	IPC brut	Ratio	%déf. brut	Ratio	%déf. brut
C	0.10	1.00	0.11	1.00	1.39	1.00	1.00	0.87	1.00	5.82	1.00	3.74
PHP	0.32	3.30	0.37	3.33	18.90	13.55	2.72	0.71	2.96	5.46	6.31	10.85
JavaScript	0.63	6.60	0.66	5.93	39.95	28.66	4.30	0.63	4.35	9.57	23.10	12.67
Python	0.73	7.69	0.86	7.77	12.65	9.08	6.22	0.69	3.60	17.73	15.77	7.66
Java	1.66	17.41	1.66	14.96	34.48	24.74	8.74	0.51	8.67	27.61	76.57	7.63



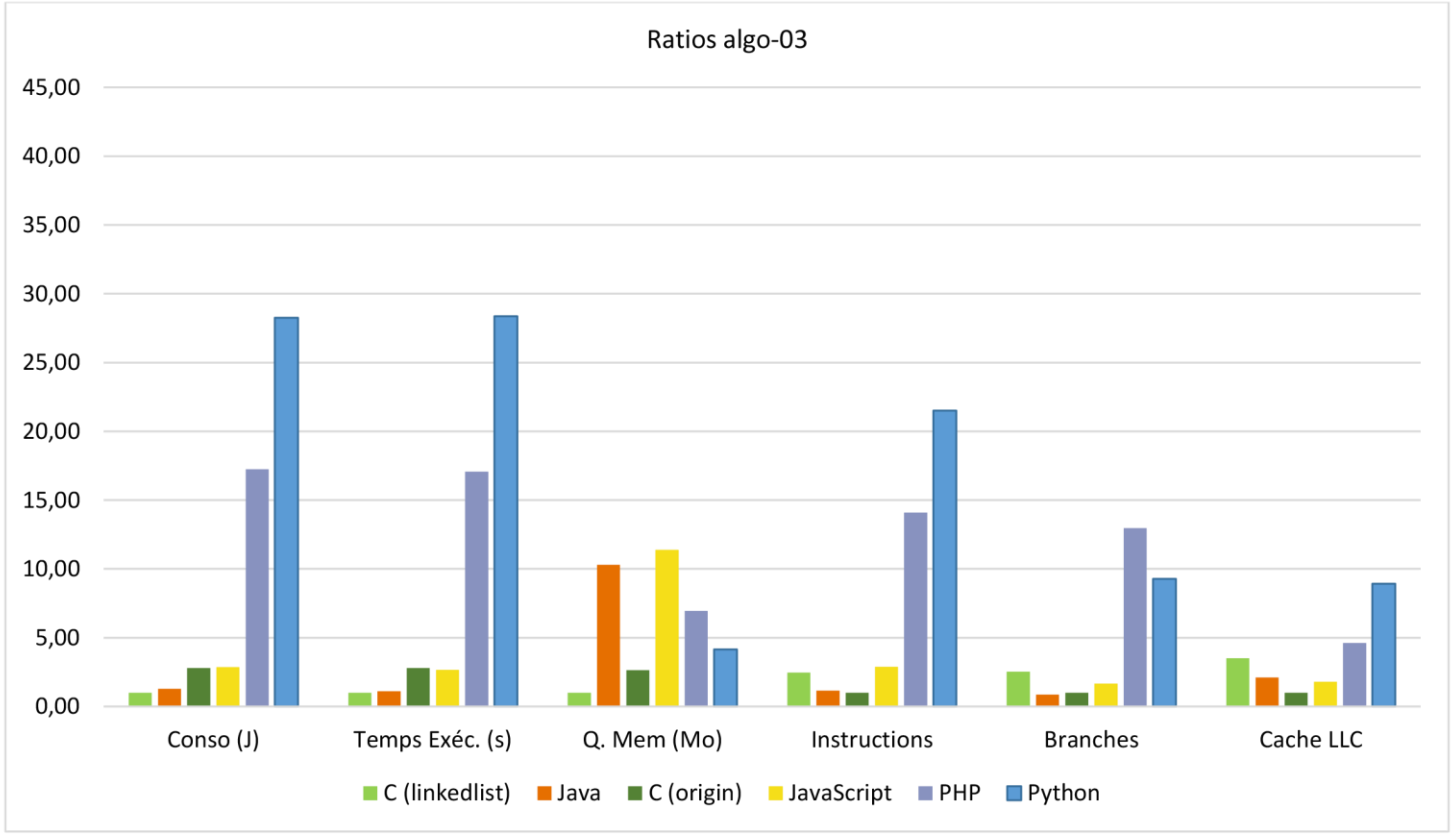
Résultats de l'exécution « Solution n°2 »

Lang.	Conso		Temps Exéc.		Q. Mémoire		Instructions		Branches		Accès Cache LLC	
	(J)	Ratio	(s)	Ratio	(Mo)	Ratio	Ratio	IPC brut	Ratio	%déf. brut	Ratio	%déf. brut
C (linkedlist)	14.13	1.00	16.15	1.00	8.55	1.00	1.00	0.90	1.00	0.30	1.00	49.77
Java	18.09	1.28	18.10	1.12	3.94	8.75	1.23	0.99	0.81	5.93	1.93	41.17
JavaScript	32.12	2.27	32.88	2.04	34.46	9.78	2.27	1.00	0.96	3.32	1.73	46.79
C (origin)	38.49	2.58	41.68	2.58	38.51	2.17	2.28	0.79	2.53	0.76	2.82	49.96
PHP	248.96	17.61	284.28	17.60	24.78	6.29	14.44	0.74	12.93	2.31	4.51	43.18
Python	407.36	28.81	466.08	28.86	14.72	3.74	21.94	0.68	9.20	33.50	6.81	33.91



Résultats de l'exécution « Solution n°3 »

Lang.	Conso		Temps Exéc.		Q. Mémoire		Instructions		Branches		Accès Cache LLC	
	(J)	Ratio	(s)	Ratio	(Mo)	Ratio	Ratio	IPC brut	Ratio	%déf. brut	Ratio	%déf. brut
C (linkedlist)	14.46	1.00	16.60	1.00	3.96	1.00	1.00	0.89	1.00	0.73	1.00	49.92
Java	18.71	1.29	18.35	1.11	40.76	10.30	1.15	0.93	0.87	6.69	2.10	38.28
C (origin)	40.45	2.80	46.44	2.80	10.40	2.63	2.46	0.79	2.52	0.32	3.51	49.67
JavaScript	41.44	2.87	44.09	2.66	45.04	11.39	2.88	0.97	1.66	2.28	1.79	45.33
PHP	249.35	17.24	283.35	17.07	27.51	6.95	14.08	0.74	12.97	2.38	4.62	42.04
Python	408.64	28.26	470.85	28.37	16.37	4.14	21.51	0.68	9.28	33.39	8.92	27.80



Comment rendre le programme plus efficient ?

- Piste d’améliorations :

Comment rendre le programme plus efficient ?

- Piste d'améliorations :
  - Stocker le nombre de pixels à colorier par lignes, colonnes
    ➔ ne pas parcourir les lignes / colonnes inutiles

Comment rendre le programme plus efficient ?

- Piste d'améliorations :

  – Stocker le nombre de pixels à colorier par lignes, colonnes
    ➔ ne pas parcourir les lignes / colonnes inutiles

Implémentation JavaScript : 44s → 85s

Comment rendre le programme plus efficient ?

- Piste d'améliorations :
  - Stocker le nombre de pixels à colorier par lignes, colonnes
    ➔ ne pas parcourir les lignes / colonnes inutiles

    Implémentation JavaScript : 44s → 85s
  - Regrouper les pixels par index d'isolement (dans un tableau)
    ➔ ne pas parcourir 7 x 480 000 pixels pour trouver ceux à remplir

Comment rendre le programme plus efficient ?

- Piste d'améliorations :
  - Stocker le nombre de pixels à colorier par lignes, colonnes
    ➔ ne pas parcourir les lignes / colonnes inutiles

    Implémentation JavaScript : 44s → 85s
  - Regrouper les pixels par index d'isolement (dans un tableau)
    ➔ ne pas parcourir 7 x 480 000 pixels pour trouver ceux à remplir

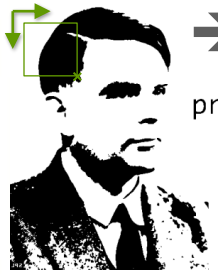
    Implémentation JavaScript : 44s → 50s

Comment rendre le programme plus efficient ?

- Piste d'améliorations :
  - Stocker le nombre de pixels à colorier par lignes, colonnes
    ➔ ne pas parcourir les lignes / colonnes inutiles

    Implémentation JavaScript : 44s → 85s
  - Regrouper les pixels par index d'isolement (dans un tableau)
    ➔ ne pas parcourir 7 x 480 000 pixels pour trouver ceux à remplir

    Implémentation JavaScript : 44s → 50s
  - Restreindre la zone de recherche pour le pixel / zone le plus efficient
    ➔ ne pas parcourir les 84 784 pixels peignables pour chaque position



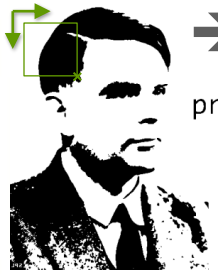
Comment rendre le programme plus efficient ?

- Piste d'améliorations :
  - Stocker le nombre de pixels à colorier par lignes, colonnes
    ➔ ne pas parcourir les lignes / colonnes inutiles

    Implémentation JavaScript : 44s → 85s
  - Regrouper les pixels par index d'isolement (dans un tableau)
    ➔ ne pas parcourir 7 x 480 000 pixels pour trouver ceux à remplir

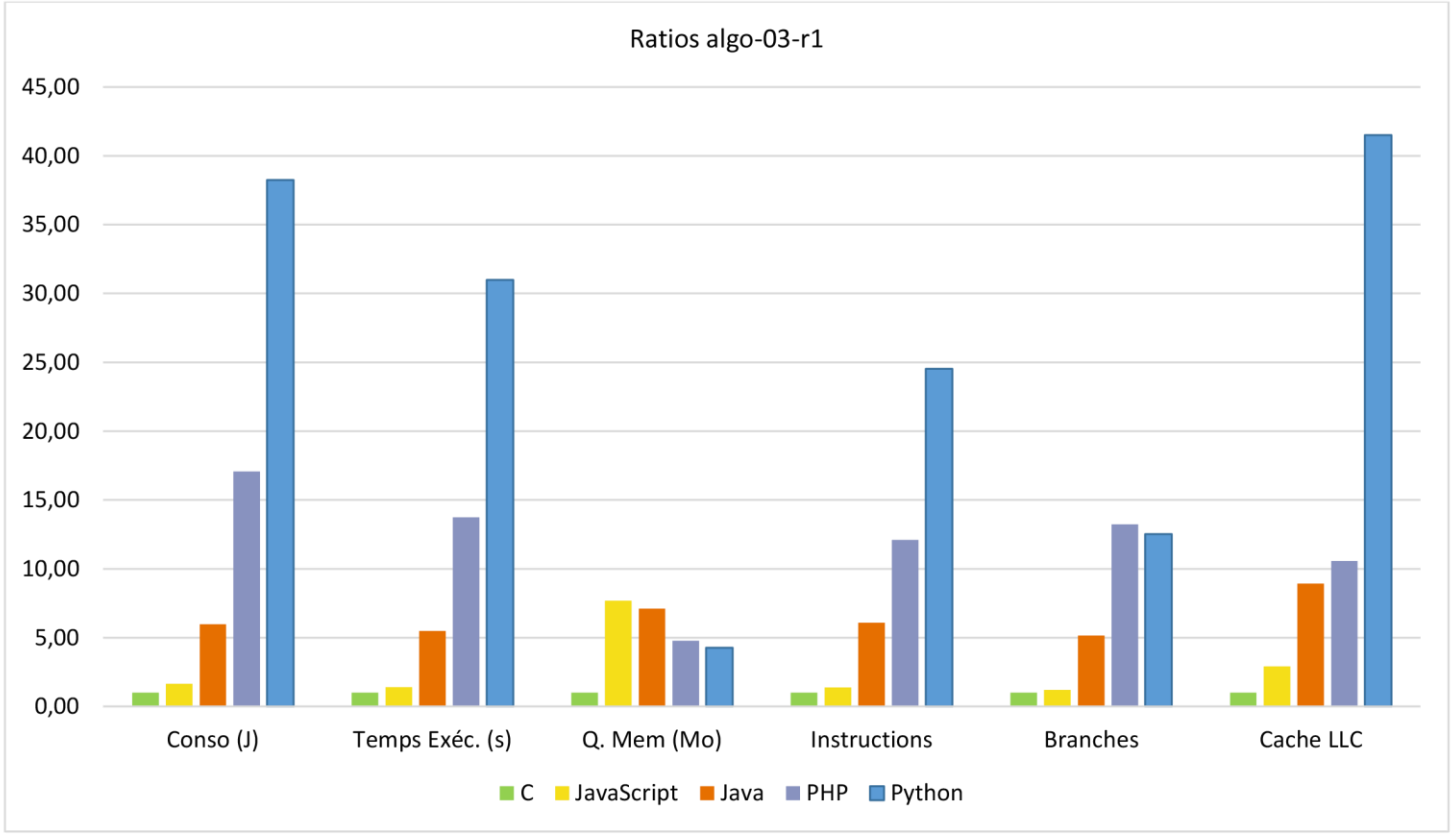
    Implémentation JavaScript : 44s → 50s
  - Restreindre la zone de recherche pour le pixel / zone le plus efficient
    ➔ ne pas parcourir les 84 784 pixels peignables pour chaque position

    Implémentation JavaScript : 44s → 5,8s



Comment rendre le programme plus efficient ?

Lang.	Conso		Temps Exéc.		Q. Mémoire		Instructions		Branches		Accès Cache LLC	
	(J)	Ratio	(s)	Ratio	(Mo)	Ratio	Ratio	IPC brut	Ratio	%déf. brut	Ratio	%déf. brut
C	2.74	1.00	4.04	1.00	5.76	1.00	1.00	0.83	1.00	1.05	1.00	12.24
JavaScript	4.51	1.65	5.62	1.39	44.22	7.68	1.38	0.83	1.19	7.26	2.91	12.26
Java	16.37	5.97	22.14	5.48	40.87	7.10	6.08	0.92	5.15	7.56	8.92	7.34
PHP	46.77	17.06	55.49	13.74	27.45	4.77	12.09	0.73	13.24	5.48	10.56	11.53
Python	104.85	38.25	125.18	30.99	24.51	4.25	24.52	0.66	12.53	26.95	41.52	6.05



Peut-on faire encore mieux ?

- Recherche inversée dans la zone « restreinte »
  ➔ on s'arrête dès qu'on est sûr de ne pas avoir de meilleure solution possible

primers.io

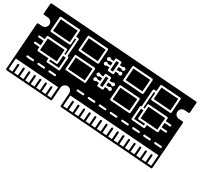
Peut-on faire encore mieux ?

- Recherche inversée dans la zone « restreinte »
  ➔ on s'arrête dès qu'on est sûr de ne pas avoir de meilleure solution possible

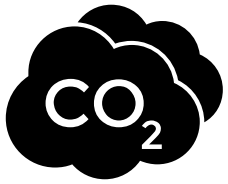
Implémentation JavaScript : 5,8s → 4,4s

Quels enseignements en tirons-nous ?

- 

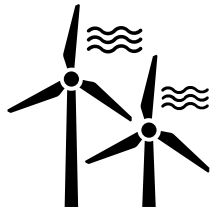
- Prendre en compte l’impact mémoire 
- Pas de recette « miracle » : expertise métier
- Impact financier : temps de dev.
- Règles d’éco-conception logicielle :
    1. Implémenter / Refactoriser
    2. Test & Profiling
    3. Réflexion

Éco-conception logicielle – 8 principes selon

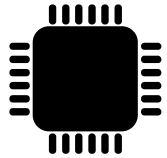


Carbone

Efficience énergétique

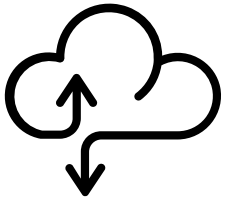


Électricité « verte »

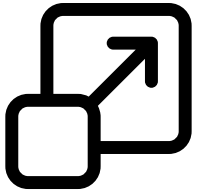


Matériel

Énergie proportionnelle



Quantité données



Scalabilité

Optimisation

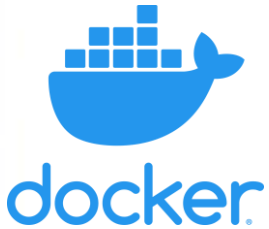
Outre les 8 principes du génie logiciel durable, il existe 2 philosophies:

- Chacun a un rôle à jouer
- La durabilité est suffisante, à elle seule, pour justifier une tâche

Bilan

- Régulation : aucune sanction -> laissé à l'appréciation des consommateurs
- Motivations tant marketing qu'économiques :
  - Réduction des dépenses énergétiques de production et/ou d'exploitation
  - Publicité sur le caractère « green » des produits
- Eco-conception basée sur le bon sens plus que sur de réelles règles

Pour faire le lien avec les autres exposés



Docker

Etude de mai 2017 : " How does Docker affect energy consumption?"

-> augmente la consommation énergétique

(Docker seul -> +2 Watts / heure)

Kubernetes

Etude de 2019 : "Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes"

-> couplé à un paramétrage correct des serveurs physiques, permet de réduire grandement la consommation énergétique

QUESTIONS

MERCI DE VOTRE ATTENTION