

# Consommation électrique d'un data center

Malo Leroy

Candidat n° 21802

2022-2023

## Ancrage au thème et motivation



- ▶ Plus de la moitié des data center français sont en zone urbaine
  - ▶ 3 % de la consommation mondiale d'électricité
  - ▶ Objectif : réduire leur consommation

# Plan

1. Modélisation d'un data center
2. Grandeurs caractéristiques
3. Puissance consommée
4. Simulation d'un data center
5. Répartition minimisant l'énergie consommée avec l'algorithme du gradient

# Modélisation d'un data center

## Différents modèles

- ▶ Modèle non retenu : ordinateur de bureau
  - ▶ Dangereux (230 V)
  - ▶ Grande inertie thermique
- ▶ Modèle retenu : Raspberry Pi
  - ▶ Peu dangereux
  - ▶ Réponse rapide aux perturbations

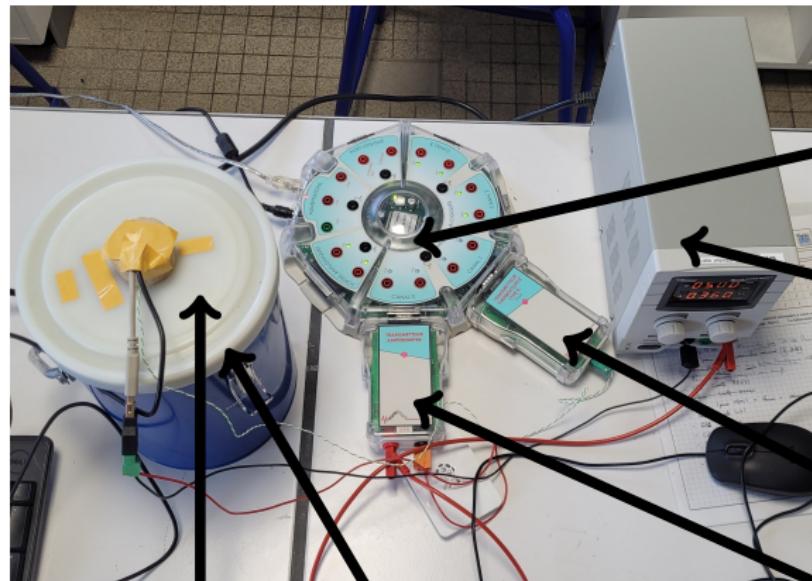


## Grandeurs caractéristiques

- ▶ Masse volumique moyenne  $\rho$  :
  - ▶ Dimensions  $85,60 \text{ mm} \times 53,98 \text{ mm} \times 17,00 \text{ mm}$
  - ▶ Masse  $90,9 \text{ g}$
  - ▶  $\rho = 1,16 \times 10^3 \text{ kg} \cdot \text{m}^{-3}$
- ▶ Capacité thermique massique moyenne  $c$
- ▶ Conductivité thermique moyenne  $\lambda$

# Grandeurs caractéristiques

## Capacité thermique



Calorimètre    Raspberry Pi (dans le calorimètre)    Ampèremètre

Carte d'acquisition

Alimentation

Thermocouple

# Grandeurs caractéristiques

## Capacité thermique

0. Mesure de la masse en eau du calorimètre  $m_{\text{calo}} = 37 \text{ g}$
1. Avant calculs ( $19,0^\circ\text{C}$ ) intensité moyenne de  $288 \text{ mA}$
2. Lancement des calculs à 1 min
3. Fin des calculs à 6 min. Pendant les calculs, on a
  - ▶ tension  $5,0 \text{ V}$
  - ▶ intensité moyenne  $381 \text{ mA}$
  - ▶ travail électrique  $612 \text{ J}$
4. Thermalisation : sur les 100 dernières secondes,  $20,6^\circ\text{C}$

(1<sup>er</sup> principe)

$$\Delta U = W = C \Delta T$$

- ▶ Capacité thermique  
 $C = 204 \text{ J/K}$
- ▶ Capacité thermique massique  
 $c = 4,5 \text{ kJ} \cdot \text{K}^{-1} \cdot \text{kg}^{-1}$
- ▶ Valeur surévaluée car on a négligé les pertes

# Grandeurs caractéristiques

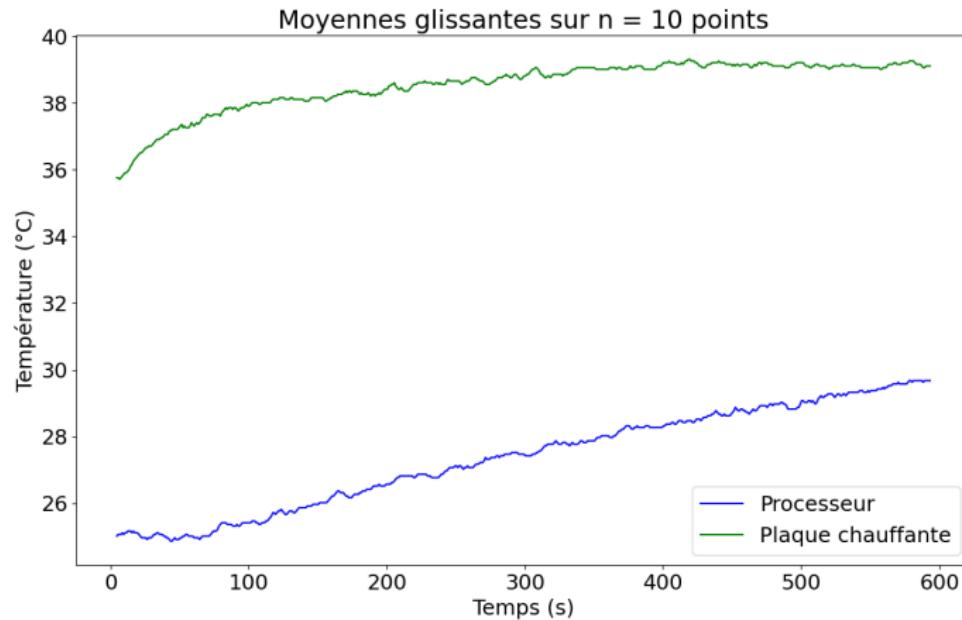
## Conductivité moyenne

### Protocole

1. Plaque chauffante
2. Mesure de la température aux deux extrémités
3. Diffusion thermique : lien entre les températures et la conductivité

# Grandeurs caractéristiques

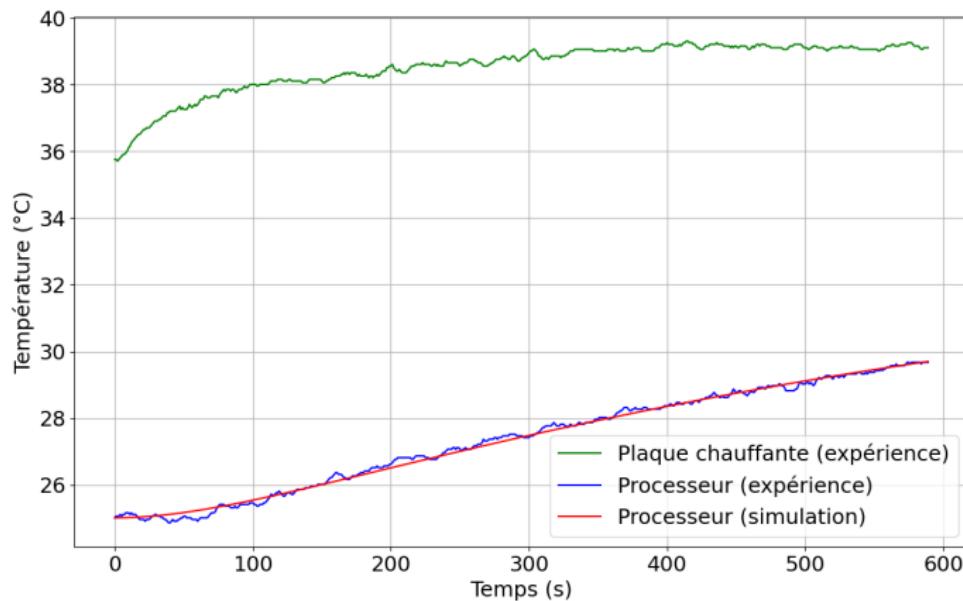
## Conductivité moyenne



$$\sigma = \frac{\Delta T}{\sqrt{3}\sqrt{n}} = 0,09 \text{ } ^\circ\text{C}$$

# Grandeurs caractéristiques

## Conductivité moyenne



On détermine  $D$  grâce à l'équation de la diffusion

$$D = 8,25 \times 10^{-8} \text{ m}^2/\text{s} \quad \lambda = D\rho c = 4,30 \times 10^{-4} \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$$

## Puissance consommée

1. Définition de la quantité de calcul
2. Mesure de la consommation du Raspberry Pi
3. Régression linéaire

# Puissance consommée

## Définition de la quantité de calcul

$K$  est le nombre de calculs par seconde

```
from time import time, sleep

def calculs(n):
    a, b = 60986.5150141834, 2831540.2372984355 # arbitraires
    for i in range(n):
        c = a ** (-b)

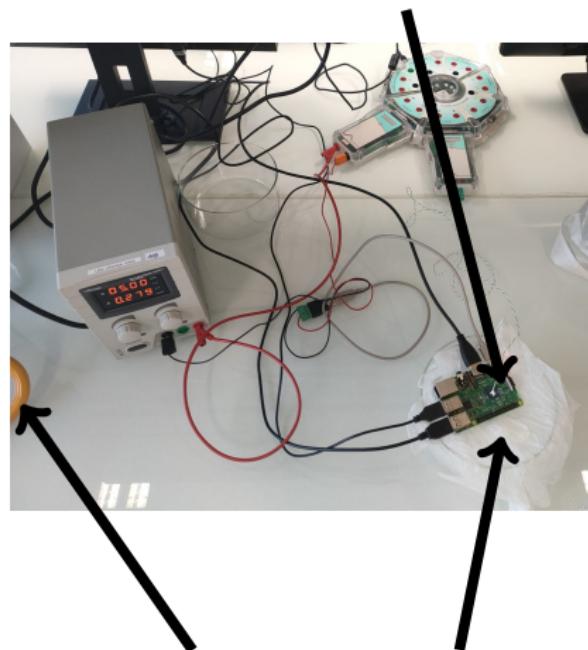
K = 100_000 # calculs par seconde
while True:
    t_i = time()
    calculs(K)
    sleep(1 -(time() -t_i))
```

# Puissance consommée

## Dispositif expérimental

Raspberry Pi

1. On impose les conditions extérieures en température
2. On impose la quantité de calculs
3. On attend le régime stationnaire
4. On mesure la température au processeur et l'intensité moyenne consommée

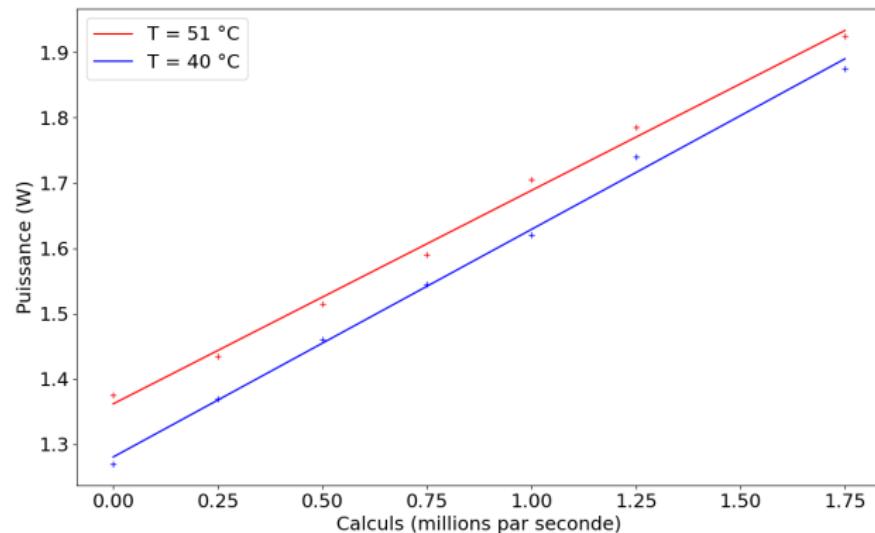


Lampe de chantier

Glace

# Puissance consommée

## Résultats expérimentaux



$$P = a \times K + b(T)$$

# Puissance consommée

## Régression linéaire

- ▶ Allure de plan

$$P = a \times T + b \times K + c$$

- ▶ Régression linéaire avec `np.linalg.lstsq`

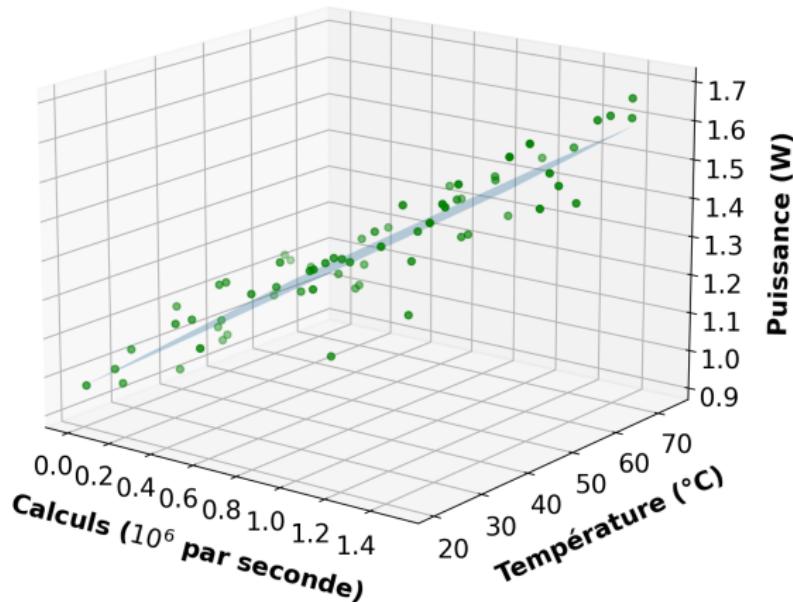
- ▶  $a = 1,8 \times 10^{-3} \text{ W/K}$

- ▶  $b = 3,6 \times 10^{-1} \text{ W/millions de calculs par seconde}$

- ▶  $c = 0,94 \text{ W}$

# Puissance consommée

## Régression linéaire



Moyenne des écarts relatifs : 3 %

# Simulation d'un data center

## Hypothèses

- ▶ Carcasse de l'ordinateur : pavé uniforme
- ▶ Source thermique : pavé centré sur la carcasse
  - ▶ Puissance volumique uniforme
- ▶ Air ambiant
  - ▶ On néglige la convection (diffusion seulement)
- ▶ Salle isolée

# Simulation d'un data center

## Principe de l'algorithme

- ▶ Temps et espace (1D) discrétilisés
- ▶ Température : tableau numpy  $T[x, t]$
- ▶ Équation aux dérivées partielles

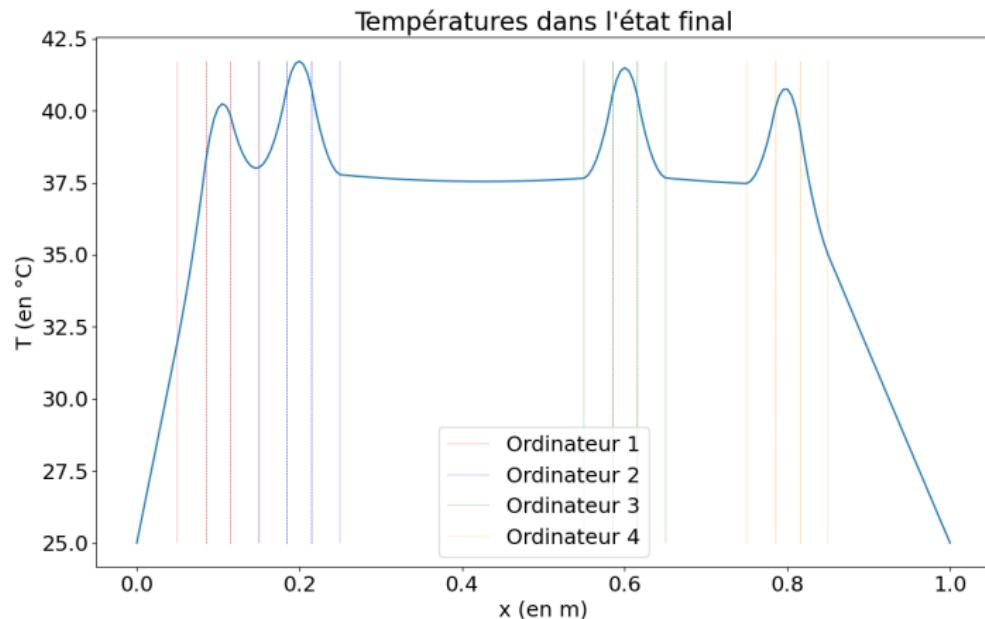
$$\frac{\partial T}{\partial t} = D(x) \frac{\partial^2 T}{\partial x^2} + P_c(x) \quad \text{où} \quad P_c(x) = \frac{P_v}{\rho c}$$

- ▶ Traduction informatique

```
T[xi, ti+1] =T[xi, ti] +(t_e / (x_e**2)) *D(xi)
    (T[xi+1, ti] -T[xi, ti] +T[xi-1, ti])
    + P_c(xi) *t_e
```

# Répartition optimale

Ordinateurs multiples et dépendances



# Répartition optimale

## Algorithme du gradient

Pour trouver un minimum d'une fonction  $f$ , ici l'énergie totale consommée en 12 h :

- ▶ Calcul du gradient au point  $M$ ,  $\nabla f(M)$
- ▶ Test d'arrêt : fin si  $||\nabla f(M)|| < \varepsilon$
- ▶ Nouveau point  $M \leftarrow M - \alpha \times \nabla f(M)$

On note  $L$  la longueur de la pièce et  $K$  la quantité de calculs totale.

Coordonnées réduites :  $\frac{x_i}{L}$  et  $\frac{K_i}{K}$  pour avoir  $f : [0, 1]^{2n} \rightarrow \mathbb{R}$

# Répartition optimale

## Algorithme du gradient

```
point =point_initial
while True:
    nablaf =gradient(f, pas, point)
    if norme(nablaf) <epsilon:
        break
    point =point -pas *nablaf
print(point) # point final
```

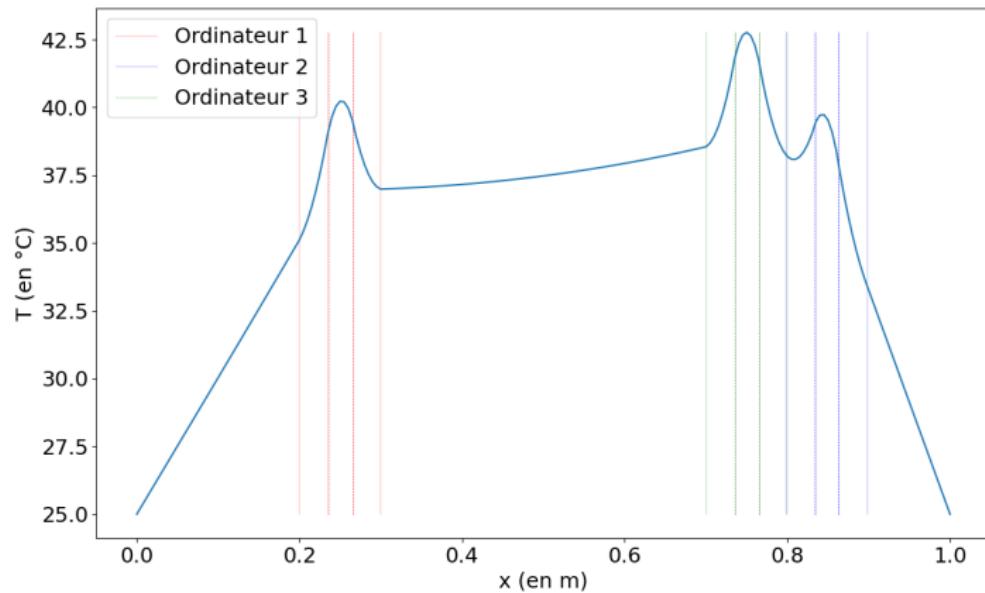
# Répartition optimale

## Résultats qualitatifs

- ▶ Éviter de tout concentrer au centre
- ▶ Donner autant de calculs à chaque groupe d'ordinateurs
- ▶ Dans chaque groupe, l'ordinateur le plus proche du mur a plus de calculs

# Répartition optimale

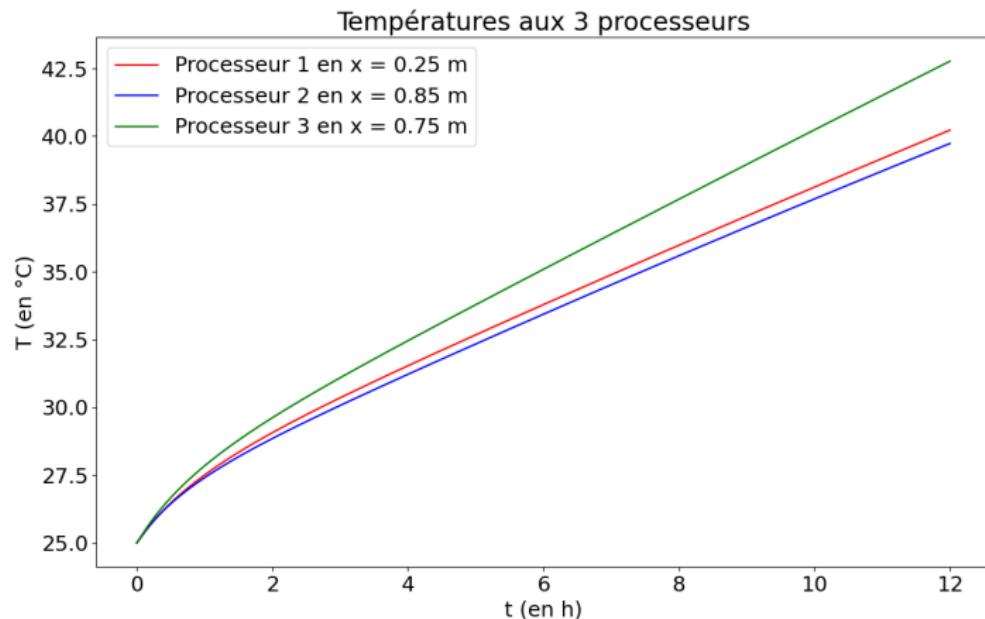
## Configurations optimales



Après 12 h de calculs

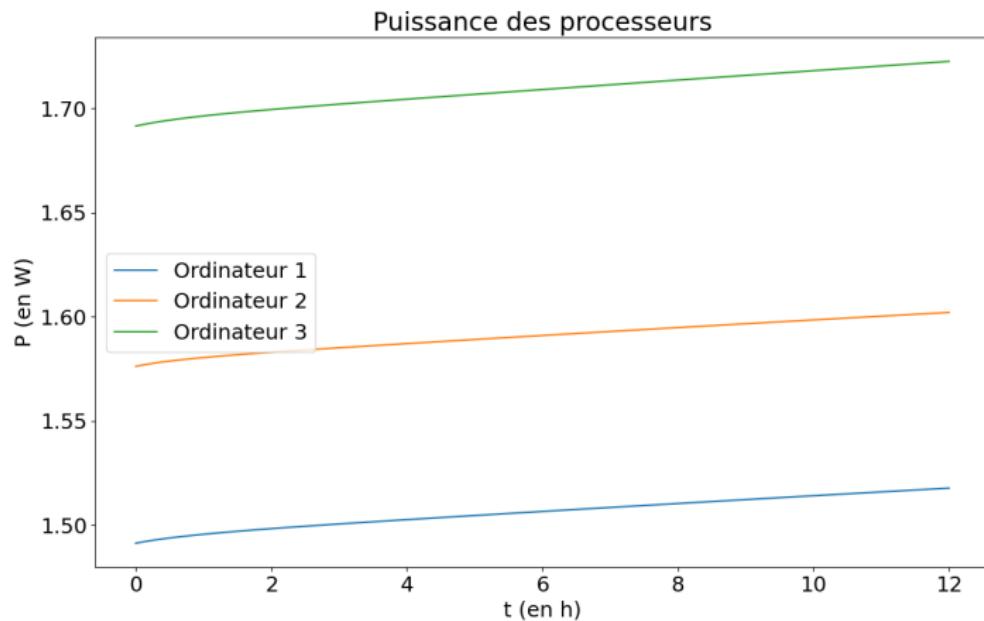
# Répartition optimale

## Configurations optimales



# Répartition optimale

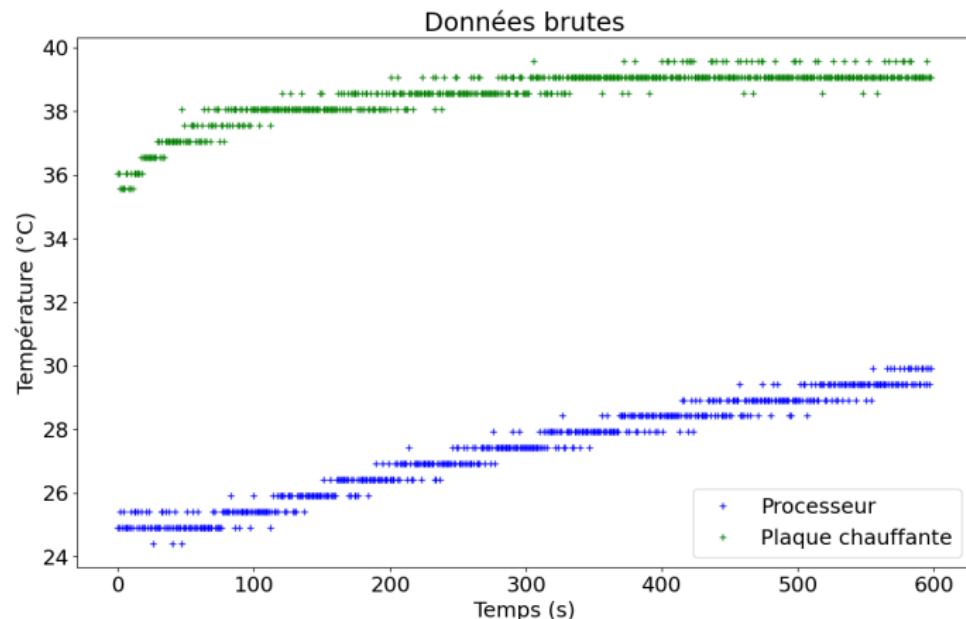
## Configurations optimales



Énergie consommée en 12 h : 20,8 kJ soit 0,06 kWh

# Conductivité moyenne

## Données brutes



# Conductivité moyenne

## Écart-type par Monte-Carlo

