TP
Les traitements informatiques et leurs consommations

CHARMEAU Antoine ARMANET Nathan

2025 - 2026

1 Equipe

Nom des étudiants avec les e-mails et numéros étudiants :

Albin Martin : albin.martin@etu.univ-lyon1.fr

2 Préambule

L'objectif de ce TP est de quantifier l'impact des traitements informatiques sur les machines. (PC, serveur, ...).

2.1 Définition

Le watt (W) est une unité internationale de mesure de la puissance énergétique : énergie produite ou consommée par unité de temps. Il existe deux méthodes de calcul :

$$Watt = \frac{Joules}{Secondes} = Volt \times Ampere$$

Un wattheure (Wh) est une unité physique qui correspond à lénergie consommée ou délivrée par un système dune puissance de 1 Watt fonctionnant pendant une heure.

3 Première mesure

Dans un premier temps, nous allons mesurer la consommation d'un ordinateur lorsqu'il est éteint et lorsqu'il est allumé.

Constater la consommation du PC :

43 Watt
10 Watt en veille, 20-40 si allumé

4 En réalisant un premier traitement

Ensuite, dans le langage de votre choix, vous écrirez un ou plusieurs programmes permettant de réaliser des opérations coûteuses. [1] :

Choix du langage 1:

python

- en RAM (insertion de nombreuse variables dans un tableau)
- en CPU (calcul mathématiques long et complexe)
- en HDD (création ; modification et suppression de fichiers)
- sur le Réseau (récupération d'un gros fichier sur internet)

Compilez les résultats obtenus dans le tableau ci-dessous :

Etape	Comsommation en (W)	Augmentation (%)
Actif	33	—
RAM	42	
CPU	84	
HDD	35	
Réseau	34	

Que concluez-vous ? Qu'est-ce qui consomme beaucoup et qu'est-ce qui consomme peu ?

Le cpu et la ram peuvent considérablement augmenter la consommation

[1] Si vous réalisez le TP sur un PC portable, vous pourrez constater des résultats plus difficiles à interpréter à cause de la batterie.

5 Refaire l'excercie dans un autre langage de programmation

Choix du langage 2:

c

Compilez les résultats obtenus dans le tableau ci-dessous :

Etape	Comsommation en (W)	Augmentation (%)
Actif	30	—
RAM	41	
CPU	65	
HDD	33	
Réseau	31	

Constatez-vous une différence de consommation avec le premier langage ? Si oui, pourquoi ?

La consommation est lègerement moindre, cela est du au fait que le C est plus optimisé que python.

6 Création d'une API

6.1 Méthodologie

Nous définissons arbitrairement une tâche comme étant le traitement de x requêtes. Chaque requête au serveur consiste en un ensemble de traitements :

- désérialisation dune requête HTTP (données en JSON)
- insertion dans une base de données du contenu de la requête
- lecture dune information dans la base de données (entrainant plusieurs) requêtes de lecture) avec un ORM
- calcul sur cette donnée
- formatage de la donnée en JSON
- retour de la donnée dans la requête HTTP

Il s'agit du traitement classique d'une API.

Nous faisons le choix de ne pas effectuer de calcul complexe, car l'usage principal des APIs consiste en la lecture et l'écriture en base de données.

La base de données est identique dans tous les tests, et elle est hébergée sur le serveur. Cela peut être discuté d'un point de vue architectural, mais cela évite les aléas de réseau entre le serveur et la base de données.

6.1.1 Protocole de mesure

Nous mesurons la consommation énergétique mesurée en watt directement sur la prise de courant du serveur.

L'objectif est de limiter au maximum les aléas de mesure "logiciel". Lors de nos tests, nous notons la consommation en watt pendant toute la durée du test, ce qui nous permet d'obtenir des Wh pour le traitement d'une tâche. Nous lançons la mesure au début de la tâche jusqu'à la fin, ce qui nous donne la durée de la tâche et sa consommation.

- Soit *ta* la tâche à effectuer.

- Soit *t* le temps pris au serveur pour répondre à l'intégralité de la tâche en seconde.

- Soit *cb* la consommation de base en w du serveur.

- Soit *c* la consommation à l'instant t du serveur.

- Soit *ctt* la consommation en watt * seconde consommée pour effectuer la tâche.

- Soit *cbt* la consommation de base en watt * seconde consommée par le serveur sur la durée de la tâche.

On a de suite : [ctt = t * c].

Dans un premier temps, nous allons étudier *ctt* en fonction des différentes technologies.

On pourrait se demander s'il est pertinent de soustraire la consommation de base *cb* du serveur, car celle-ci est supposée être la même pour chaque technologie.

Nous avons décidé en première analyse de le garder, afin de pouvoir voir la consommation *ctt* avec la *cbt* et de voir si l'impact des technologies est faible et non significatif avec la consommation de base *cb*.

6.1.2 Création de la tâche

Dans cette partie, vous devrez créer une API REST permettant d'effectuer les actions suivantes :

- GET (/api/objects) : Récupérer une liste d'objet paginé.
- GET (/api/objects/{id}) : Récupérer un objet par son ID
- POST (/api/objects) : Insérer un nouvel objet en BDD
- PUT (/api/objects/{id}) : Modifier un objet existant
- DELETE (/api/objects/{id}) : Supprimer un objet existant

Pour que les tests se rapprochent le plus possible d'une utilisation réelle, vous devrez précharger la table avec quelques données (un dump vous est mis à disposition).

L'ensemble du projet devra être livré avec un fichier docker-compose.yml comprenant deux services : l'API et la base de données.

Compilez les résultats obtenus dans le tableau ci-dessous :

	Consommation total (Wh)	Durée (s)	Traitement seul (wh)
Consommation du serveur		—	—
GET (/api/objects)			
GET (/api/objects/{id})			
POST (/api/objects)			
PUT (/api/objects/{id})			
DELETE (/api/objects/{id})			

Choix du langage et la stack technologique :

Que pouvez-vous conclure de vos observations ?

6.2 Mise en commun

Dans cette partie, nous allons mettre en commun tous les résultats des groupes afin de les comparer.

Existe-t-il des différences significatives entre les résultats des groupes ? Si oui, pourquoi ?