



**PARIS** Albin  
**YAZICI** Servan

# Sommaire

- Présentation de NodeJs
- Démonstration de Node
- Présentation de Spring
- Démonstration de Spring
- Point de vue du développeur
- Point de vue de l'utilisateur
- Conclusion

# C'est quoi NodeJS ?

NodeJs est un environnement d'exécution Javascript léger **Single-Threaded** et **Non-Blocking** .

# Quelques définitions

Single thread

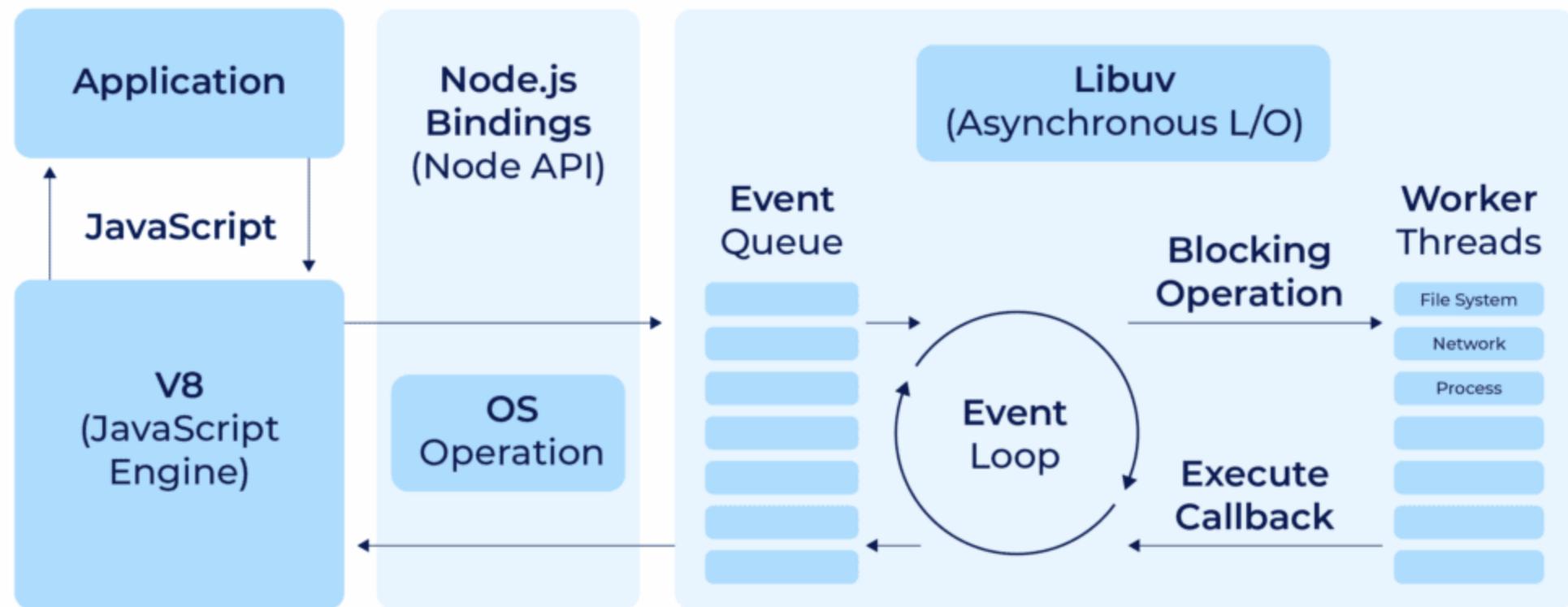
Multi-threading

Bloquant

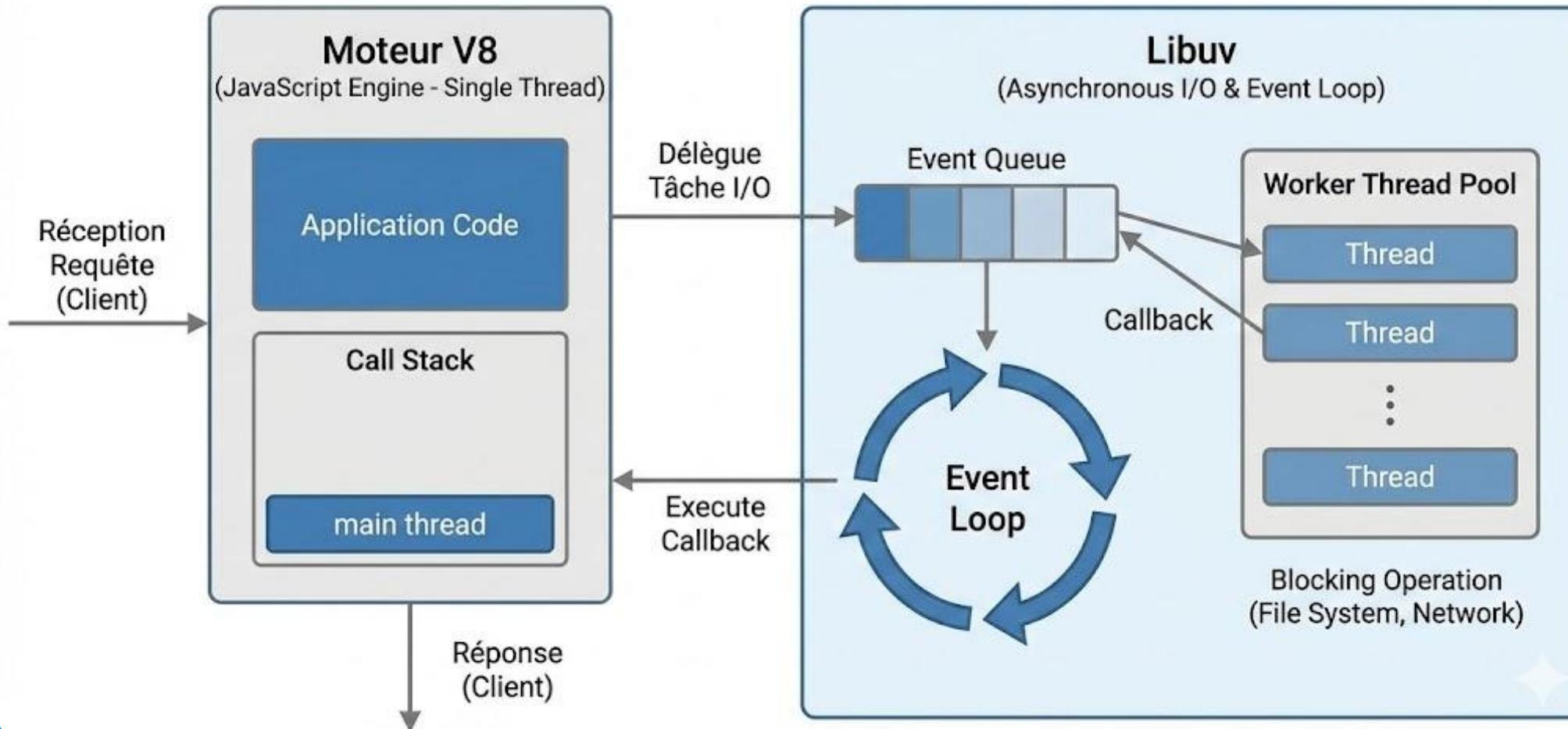
Non-bloquant

Léger

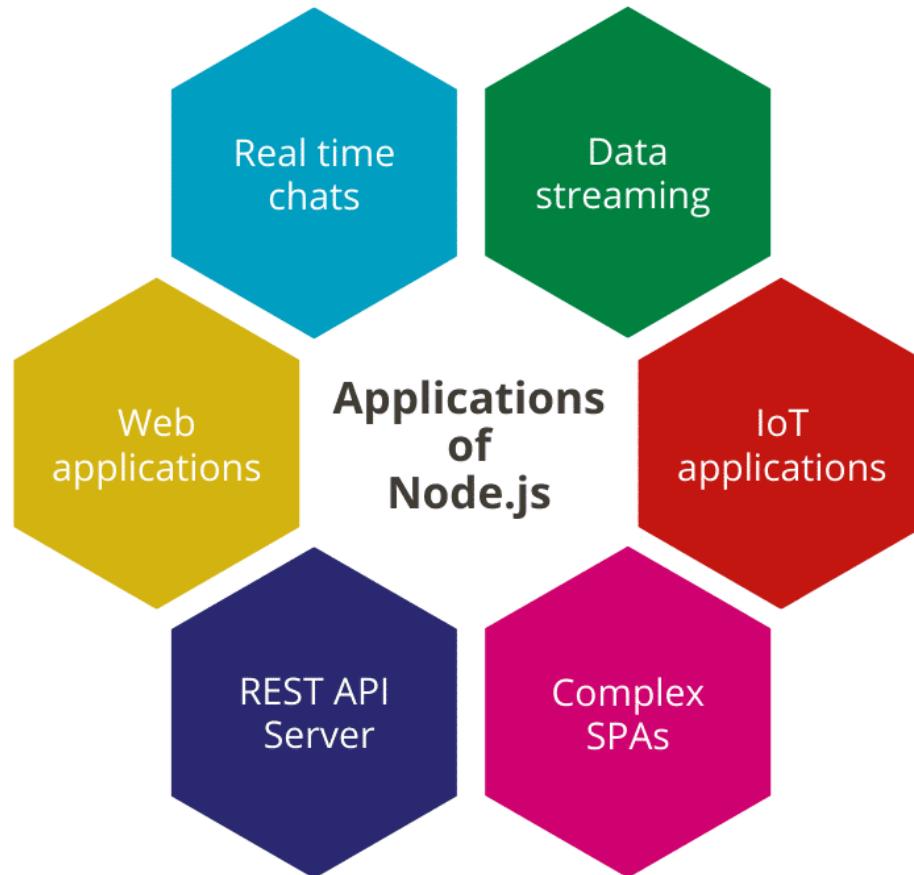
# Node.js Architecture



# Traitement d'une Requête Node.js



# Type d'application



# Demo NodeJs

Présentation :

Puissance de la librairie Libuv

Limite du single thread



# L'Écosystème Spring

---

Des fondations Java EE à l'accélération Spring Boot.

# Les Fondations (Java & Java EE)



**Java :**  
Le langage (les briques).



**Java EE (Jakarta EE) :** Les spécifications  
**Le Concept :** L'Abstraction. Séparer l'interface (Standard) de l'implémentation (Outil tiers).  
**Exemple :** JPA (Interface) vs Hibernate (Implémentation)

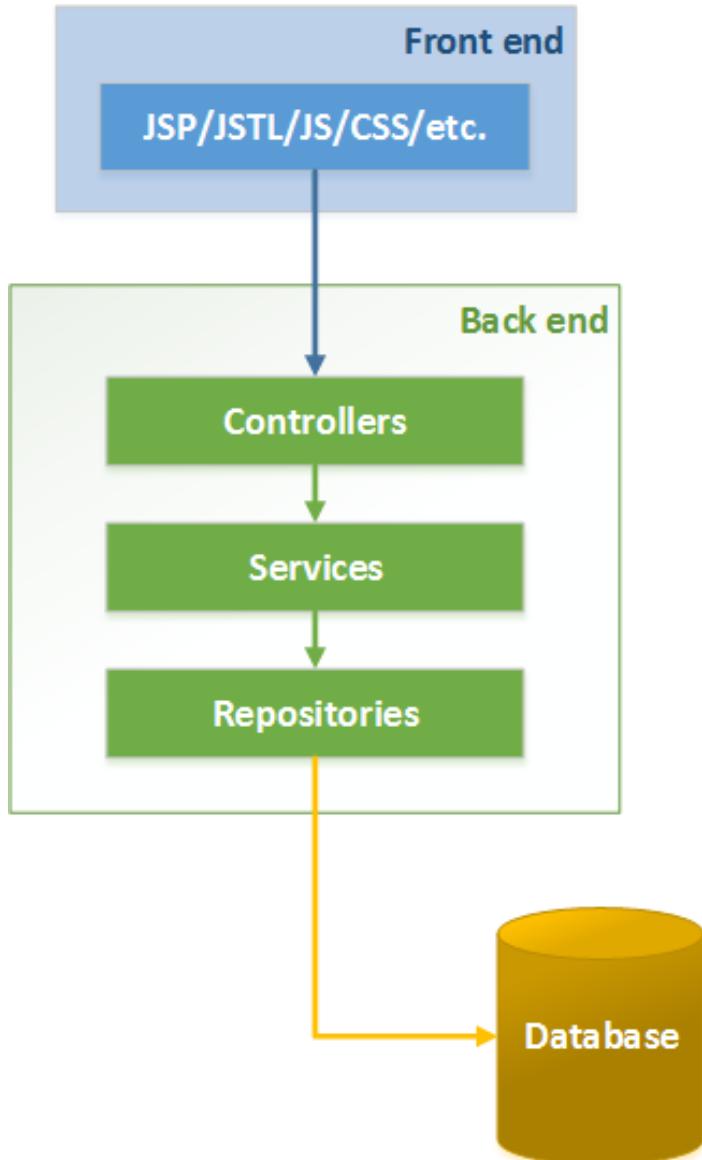
# Les problèmes historiques

```
public void saveUser(User user) {  
    Connection conn = null;  
    PreparedStatement stmt = null;  
    try {  
  
        conn = datasource.getConnection();  
  
        String sql = "INSERT INTO users (name) VALUES (?)";  
        stmt = conn.prepareStatement(sql);  
        stmt.setString(1, user.getName());  
  
        stmt.executeUpdate();  
  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        try { if (stmt != null) stmt.close(); } catch (SQLException e) {}  
        try { if (conn != null) conn.close(); } catch (SQLException e) {}  
    }  
}
```

**Code "Boilerplate"** : Verbeux et répétitif

Java EE permet d'éviter ce problème

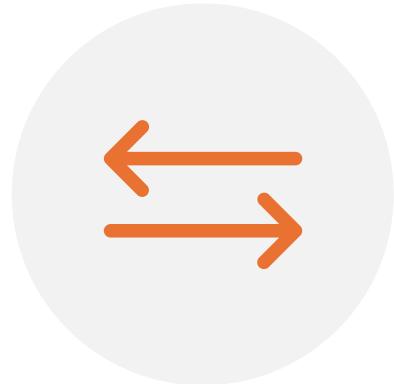
```
public void saveUser(User user) {  
    entityManager.persist(user);  
}
```



# Dépendance forte entre classes

```
new Controller(  
    new Service(  
        new Repository()  
    )  
)
```

# La Solution Spring Framework



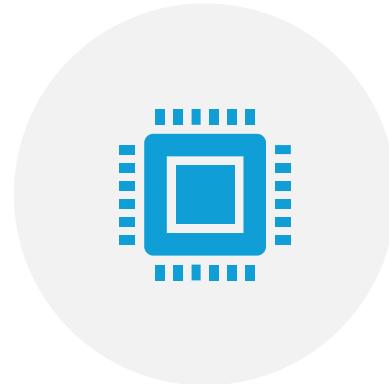
## **INVERSION DE CONTRÔLE (IOC) :**

LE FRAMEWORK (LE CONTENEUR) GÈRE LE CYCLE DE VIE DES OBJETS.



## **INJECTION DE DÉPENDANCES (DI) :**

ASSEMBLAGE AUTOMATIQUE DES COMPOSANTS (@AUTOWIRED, CONSTRUCTEUR).



## **SCOPE SINGLETON :**

UNE SEULE INSTANCE PAR COMPOSANT (PERFORMANCE MÉMOIRE).

# Les problèmes non résolus

**Dependency Hell** : Conflits de versions entre librairies (Maven).

**Configuration Lourde** : classes `@Configuration` manuelles.

**Déploiement complexe** : Installation obligatoire d'un serveur externe (Tomcat, Wildfly).

# L'Accélérateur Spring Boot

Objectif : Déploiement un serveur en 5 min

- Serveur embarqué Tomcat
- Mise en place de Starter au niveau des dépendances
- Auto-configuration

# Exemple de starter

```
<dependency>
  <groupId>org.springframework.boot</groupId>
    <artifactId>
      spring-boot-starter-security
    </artifactId>
</dependency>
```

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>
      spring-boot-starter
    </artifactId>
    <version>3.3.3</version>
    <scope>compile</scope>
  </dependency>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>6.1.12</version>
    <scope>compile</scope>
  </dependency>

  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>6.3.3</version>
    <scope>compile</scope>
  </dependency>

  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>6.3.3</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

# **Demo Spring**

# Expérience développeur

Concrètement, qu'est-ce que ça change au quotidien de travailler avec l'un ou l'autre ?



# Le confort d'écriture (Language & Typage)

JavaScript



Typage dynamique



Typage statique

# Exemple

JavaScript



```
t=null  
console.log(t) -> null  
  
t += 'ing'  
console.log(t) -> 'nulling'
```

D'autres exemple sur <https://javascriptwtf.com/>

# La gestion de l'écosystème



Node Package Manager  
**Structure :** node\_modules



Maven (Spring boot)  
**Structure :** pom.xml

# **Qu'est ce qui importe chez le client ?**

# Qu'est ce qui importe chez le client ?

Un site sans bugs ?

# Qu'est ce qui importe chez le client ?

Un site sans bugs ?

Un beau site ?

# Qu'est ce qui importe chez le client ?

Un site sans bugs ?

Un beau site ?

Un site pratique ?

# Qu'est ce qui importe chez le client ?

Un site sans bugs ?

Un beau site ?

Un site pratique ?

Un site RAPIDE

# **Benchmark de performance**

# Comment tester un backend web ?

Framework de test de charge



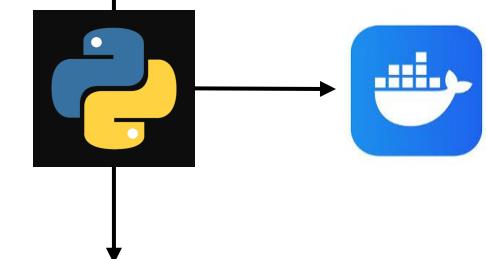
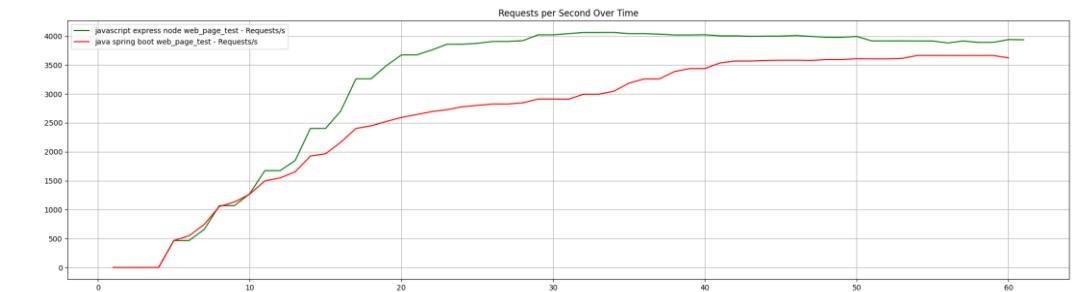
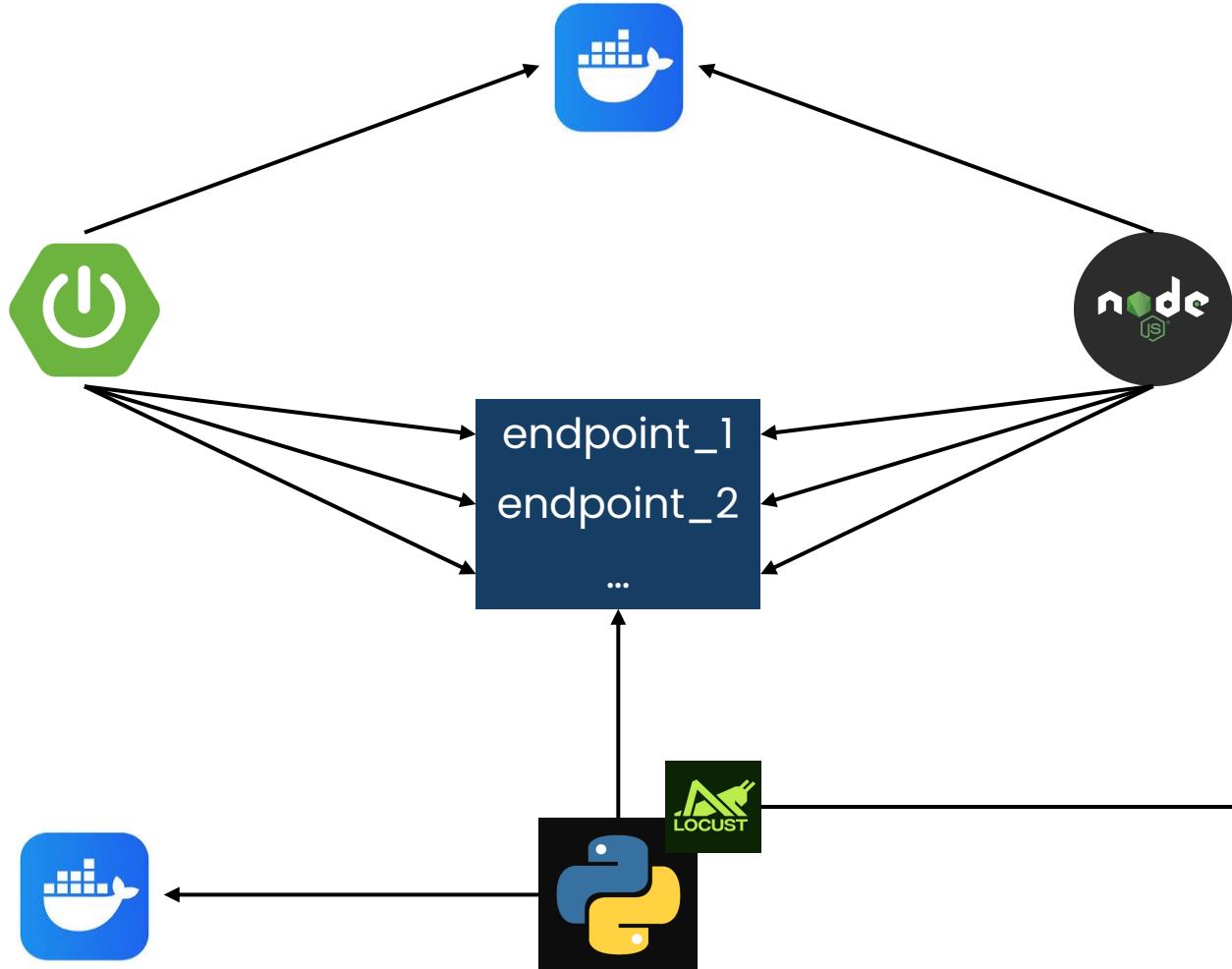
Outil de benchmark fait par la communauté



→ <https://github.com/abdelaziz-mahdy/backend-benchmark>



# Comment le benchmark fonctionne ?



# Que font nos tests ?

## **LOCUST\_RUNTIME (Temps d'exécution)**

-> 60 secondes

## **LOCUST\_USERS (Nombre d'utilisateurs total)**

-> 5000 utilisateurs

## **LOCUST\_SPAWN\_RATE (Fréquence d'apparition des utilisateurs)**

-> 83.33 utilisateurs par secondes

## **WAIT\_TIME (Temps d'attente pour un utilisateur avant de relancer sa tâche)**

-> Entre 0.1 et 1 seconde

# Que font nos tests ?

## **db\_test**

- > POST(« /notes/ ») : insère une note en BD
- > GET(« /notes/ ») : récupère les 100 premières notes

## **no\_db\_test**

- > GET(« /no\_db\_endpoint/ ») : Renvoie un OK(200)
- > GET(« /no\_db\_endpoint2/ ») : Renvoie un OK(200)

## **file\_io\_db\_test**

- > POST(« /notes/file\_io/ ») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(« /notes/file\_io/latest/ ») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

## **file\_io\_no\_db\_test**

- > GET(« /no\_db\_file\_io/ ») : Crée un fichier, écrit dedans, le lit, puis renvoie un OK(200)

## **web\_page\_test**

- > GET(« /no\_db\_file\_io/ ») : Renvoie une page HTML statique avec ces assets (CSS, JS)

## **compute\_test**

- > GET(« /compute/fibonacci/ ») : Calcule un Fibonacci de 42 et renvoie un OK(200)

## **bad\_compute\_test**

- > GET(« /compute/fibonacci/ ») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

# Exemple du « **bad** » fibonacci



```
private long recursiveFibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    return recursiveFibonacci(n - 1) + recursiveFibonacci(n - 2);
}
```

«NoteController.java»

```
@GetMapping({"/compute/fibonacci/bad", "/compute/fibonacci/bad/"})
public ResponseEntity<?> computeBadFibonacci(@RequestParam(value = "n", defaultValue = "38") int n) {
    if (n < 0 || n > 40) {
        return ResponseEntity.badRequest().body("n must be between 0 and 40");
    }
    long start = System.nanoTime();
    long result = recursiveFibonacci(n);
    long durationMicros = (System.nanoTime() - start) / 1_000;
    Map<String, Object> payload = new HashMap<>();
    payload.put("n", n);
    payload.put("result", result);
    payload.put("durationMicros", durationMicros);
    return ResponseEntity.ok(payload);
}
```

# Exemple du « **bad** » fibonacci

```
function recursiveFib(n) {
  if (n <= 1) {
    return n;
  }
  return recursiveFib(n - 1) + recursiveFib(n - 2);
}
```



«server.js»

```
app.get(['/compute/fibonacci/bad', '/compute/fibonacci/bad/'], (req, res) => {
  const n = Number.parseInt(req.query.n ?? '38', 10);
  if (Number.isNaN(n) || n < 0 || n > 40) {
    return res.status(400).send('n must be between 0 and 40');
  }

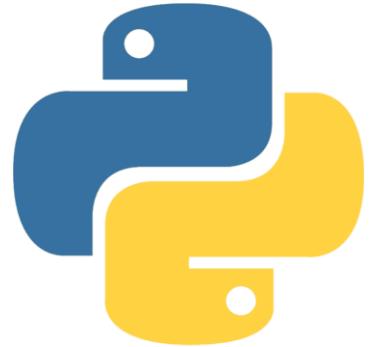
  const start = process.hrtime.bigint();
  const result = recursiveFib(n);
  const durationMicros = Number((process.hrtime.bigint() - start) / 1000n);
  res.json({ n, result, durationMicros });
});
```

# Exemple du « **bad** » fibonacci

```
from locust import FastHttpUser, task, between

class BadComputeUser(FastHttpUser):
    wait_time = between(0.1, 1)

    @task
    def compute_bad_fibonacci(self):
        self.client.get("/compute/fibonacci/bad/?n=38", name="/compute/fibonacci/bad")
```



«bad\_compute\_test.py»



# Exemple du « bon » fibonacci



```
private long fibonacci(int n) {  
    if (n == 0) {  
        return 0L;  
    }  
    if (n == 1) {  
        return 1L;  
    }  
  
    long previous = 0L;  
    long current = 1L;  
    for (int i = 2; i <= n; i++) {  
        long next = previous + current;  
        previous = current;  
        current = next;  
    }  
    return current;  
}
```

«NoteController.java»

```
@GetMapping({"/compute/fibonacci/", "/compute/fibonacci"})  
public ResponseEntity<?> computeFibonacci(@RequestParam(value = "n", defaultValue = "42") int n) {  
    if (n < 0 || n > 92) {  
        return ResponseEntity.badRequest().body("n must be between 0 and 92");  
    }  
  
    long start = System.nanoTime();  
    long result = fibonacci(n);  
    long durationMicros = (System.nanoTime() - start) / 1_000;  
  
    Map<String, Object> payload = new HashMap<>();  
    payload.put("n", n);  
    payload.put("result", result);  
    payload.put("durationMicros", durationMicros);  
    return ResponseEntity.ok(payload);  
}
```

# Exemple du « bon » fibonacci



«server.js»

```
function recursiveFib(n) {
  if (n <= 1) {
    return n;
  }
  return recursiveFib(n - 1) + recursiveFib(n - 2);
}
```

```
app.get(['/compute/fibonacci', '/compute/fibonacci/'], (req, res) => {
  const n = parseInt(req.query.n ?? '42', 10);
  if (Number.isNaN(n)) {
    return res.status(400).send('n must be a number');
  }
  if (n < 0 || n > 92) {
    return res.status(400).send('n must be between 0 and 92');
  }

  const start = process.hrtime.bigint();
  const result = fibonacci(n);
  const durationMicros = Number((process.hrtime.bigint() - start) / 1000n);

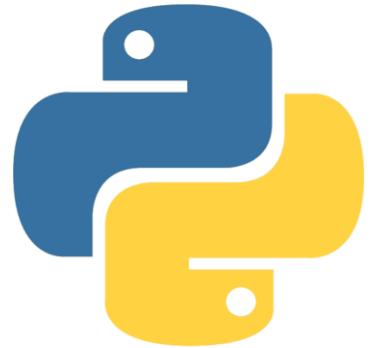
  return res.json({ n, result, durationMicros });
});
```

# Exemple du « bon » fibonacci

```
from locust import FastHttpUser, task, between

class ComputeUser(FastHttpUser):
    wait_time = between(0.1, 1)

    @task
    def compute_fibonacci(self):
        self.client.get("/compute/fibonacci/?n=42", name="/compute/fibonacci")
```



«bad\_compute\_test.py»



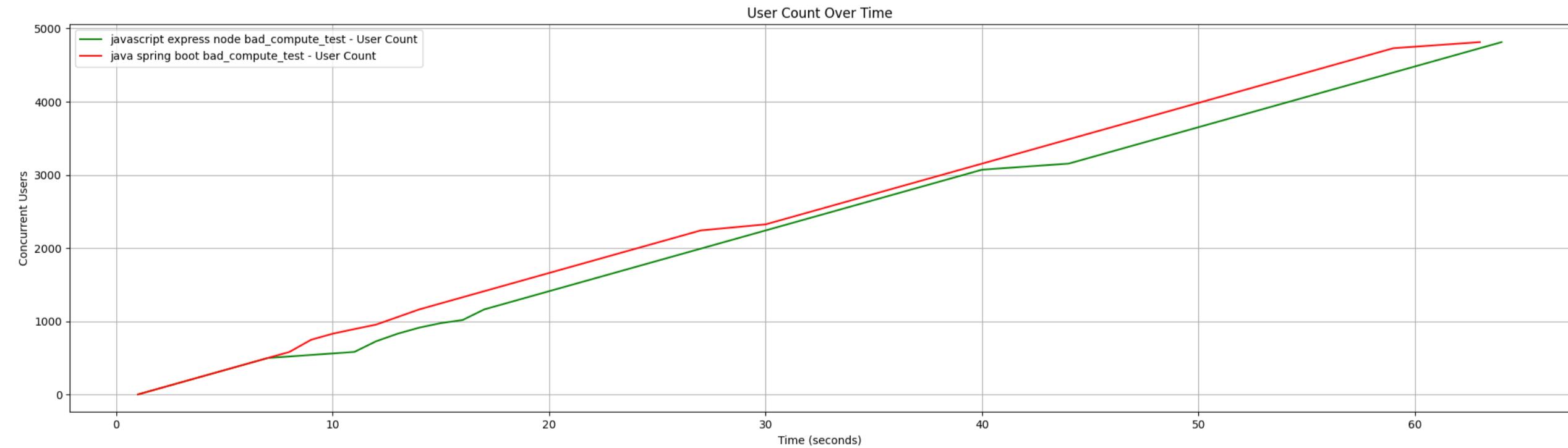
# **Observons les résultats**

# Résultats

## **bad\_compute\_test**

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

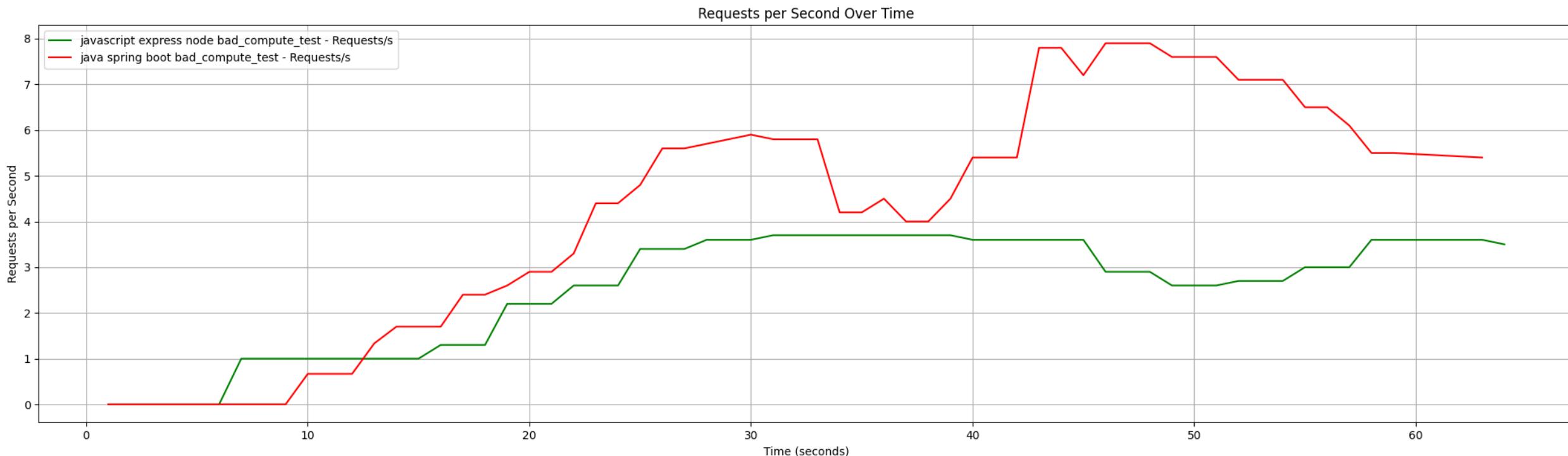
# Résultats



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

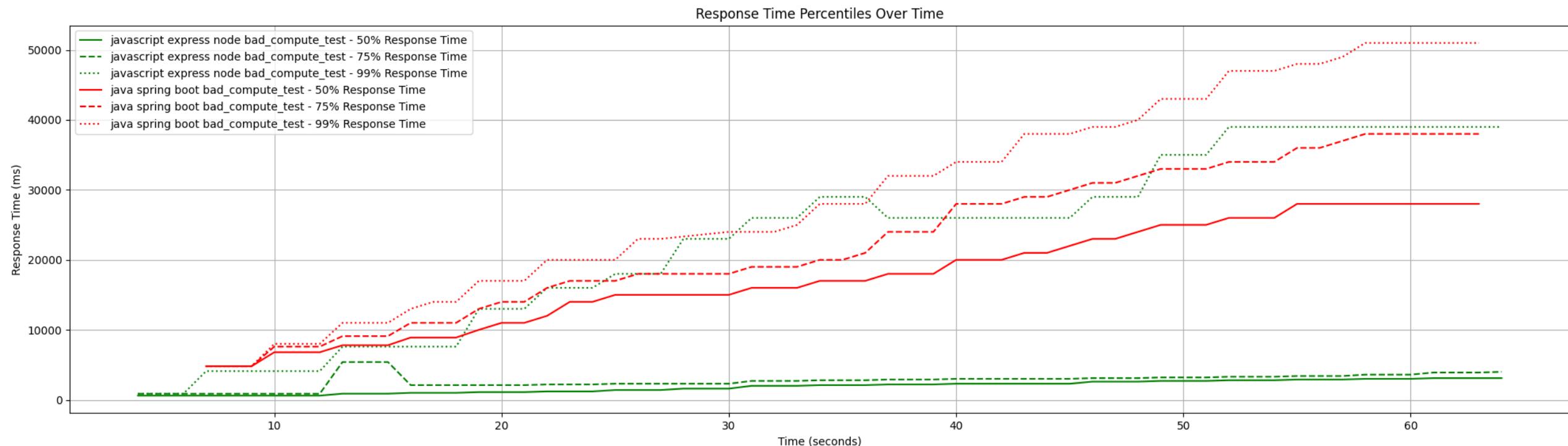
# Résultats



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

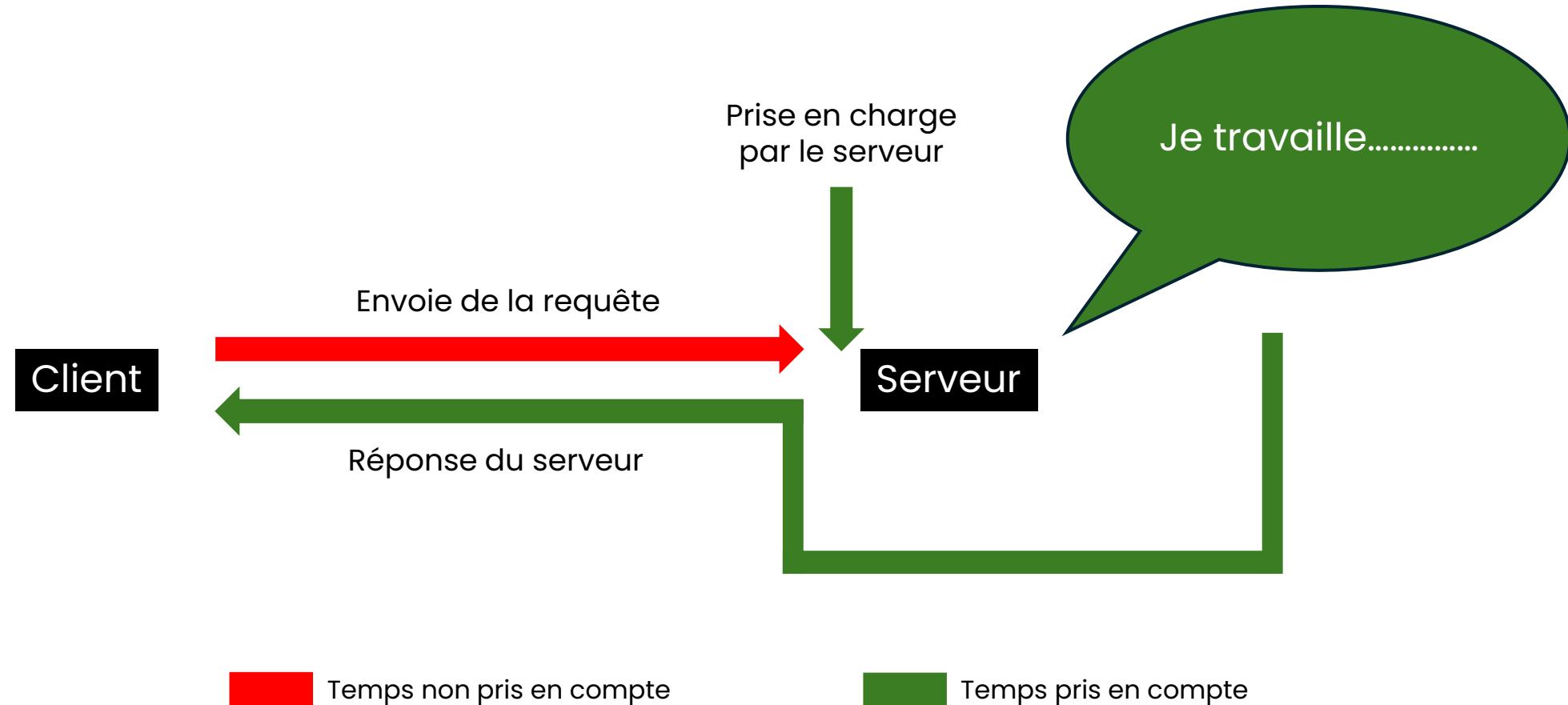
# Résultats



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

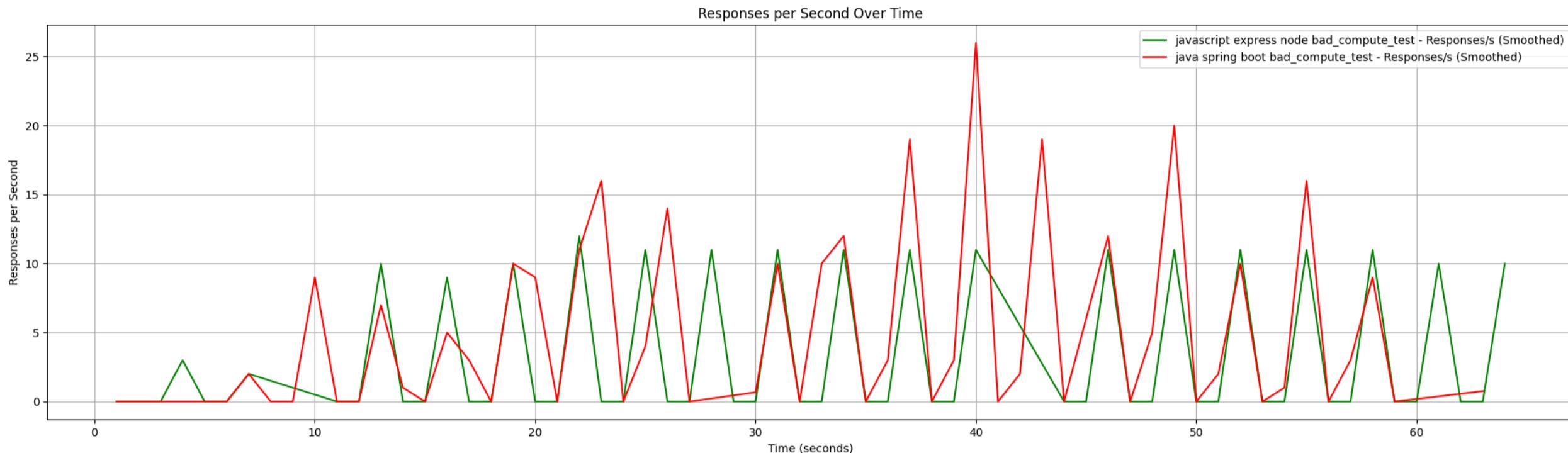
# Résultats



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

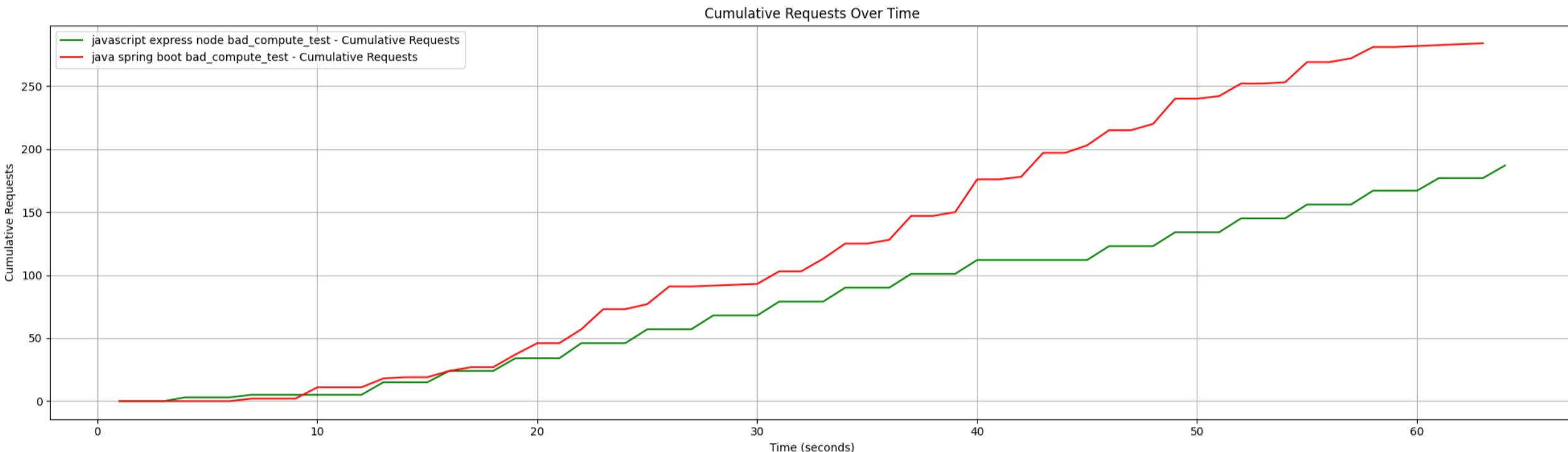
# Résultats



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

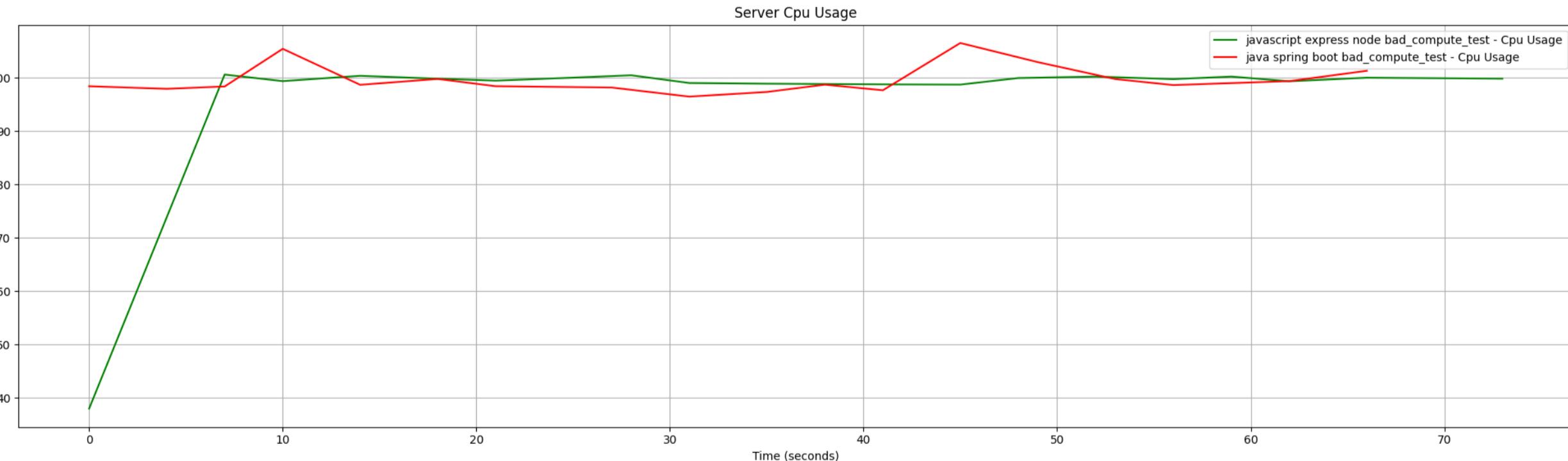
# Résultats



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

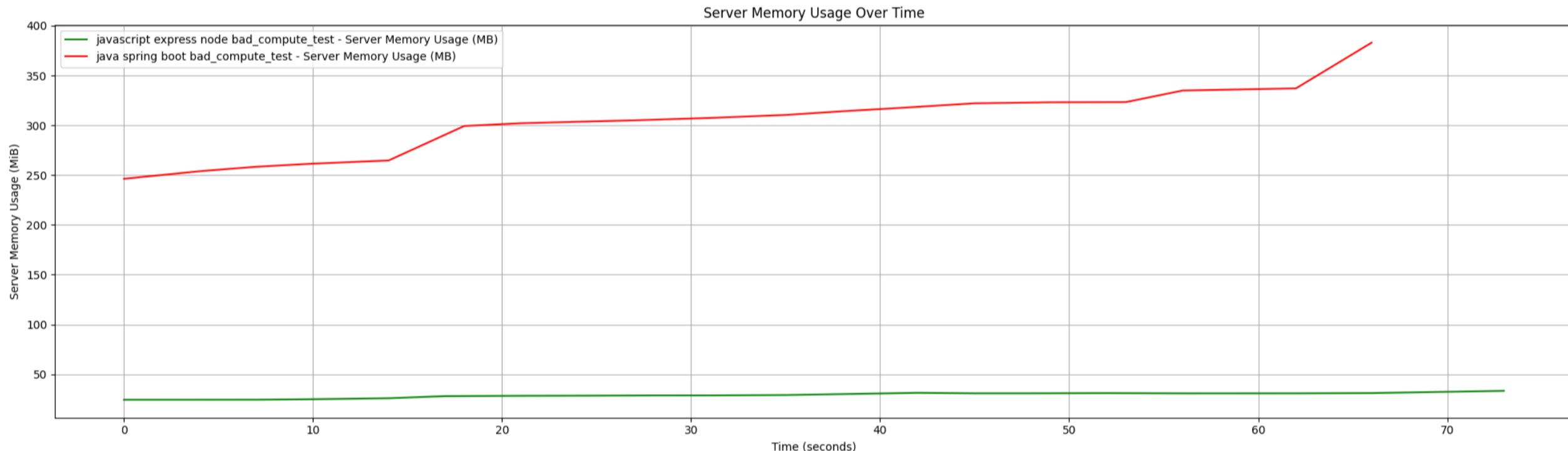
# Résultats



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

# Résultats



## bad\_compute\_test

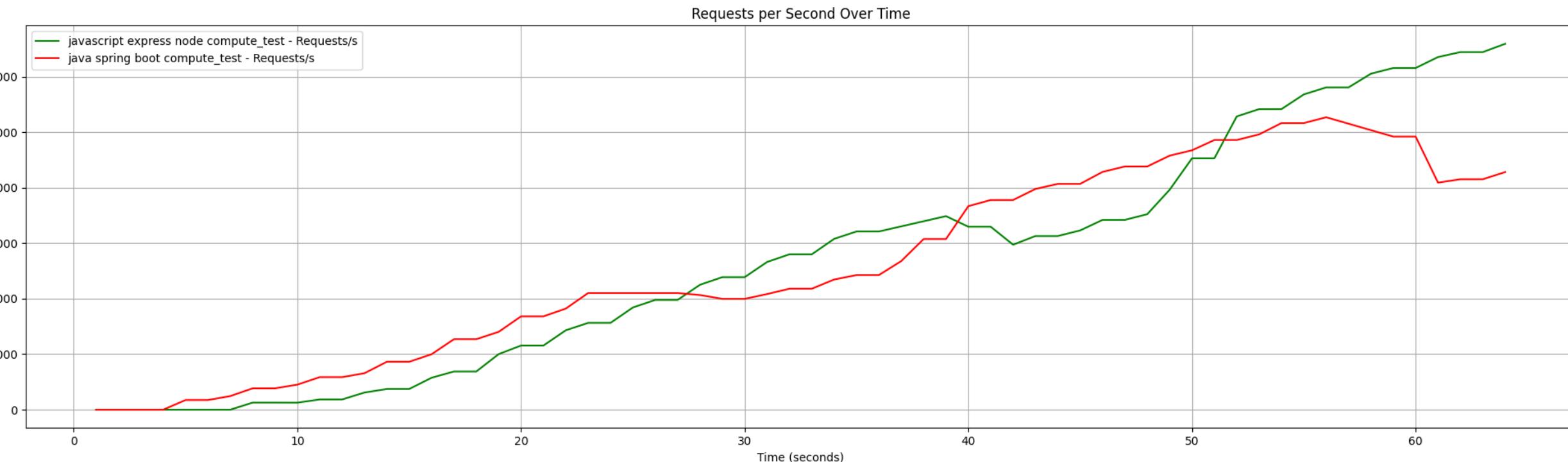
-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

# Résultats

## **compute\_test**

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

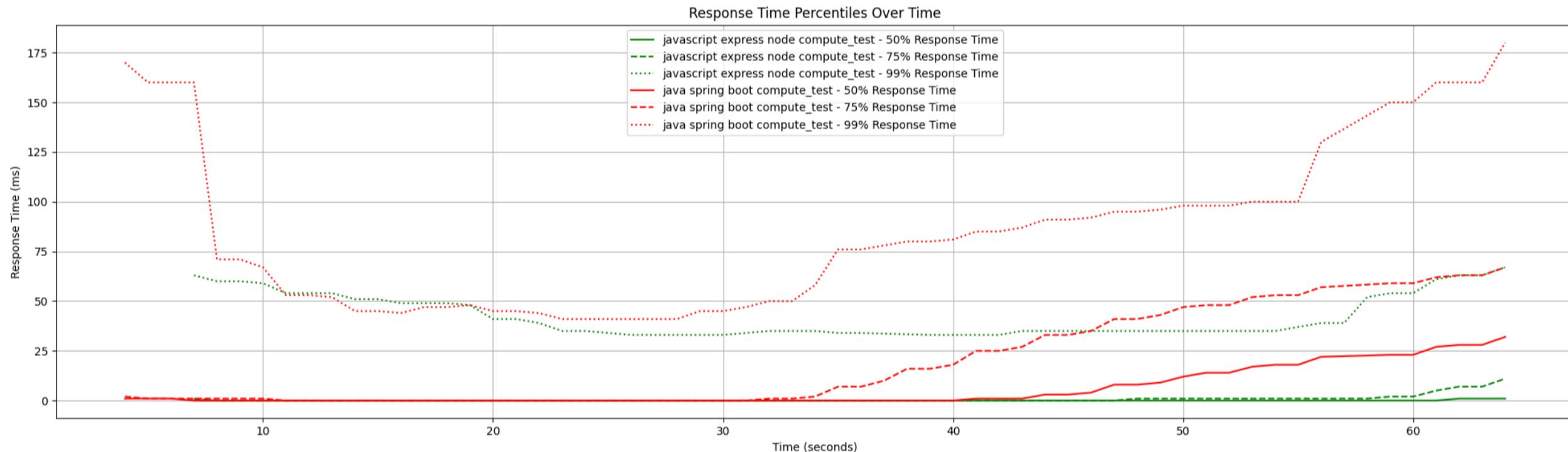
# Résultats



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

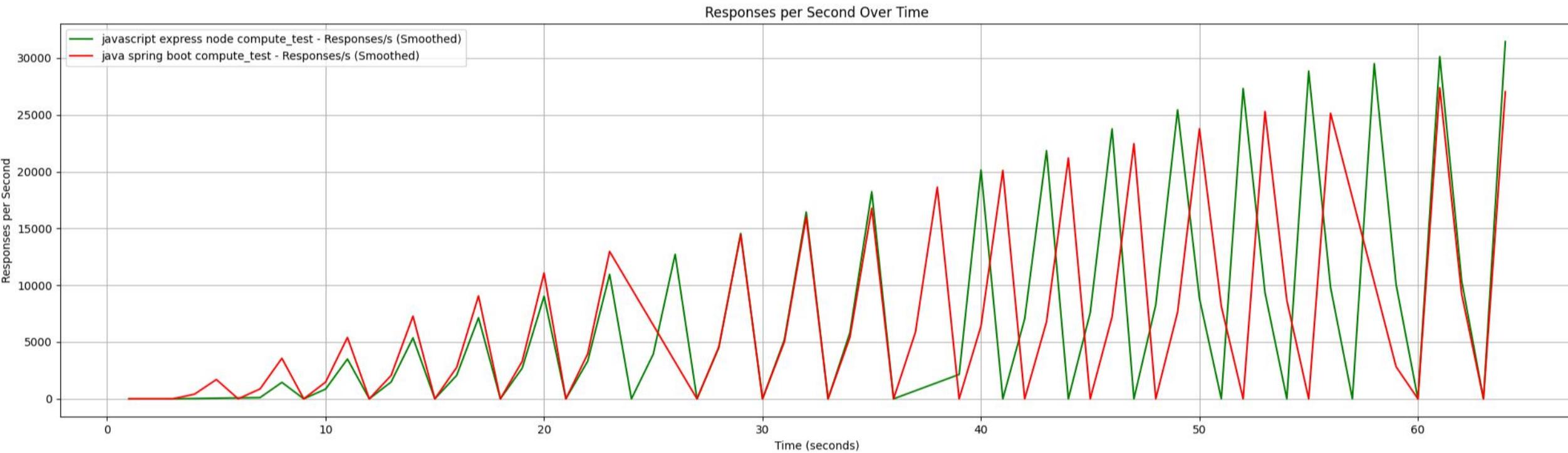
# Résultats



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

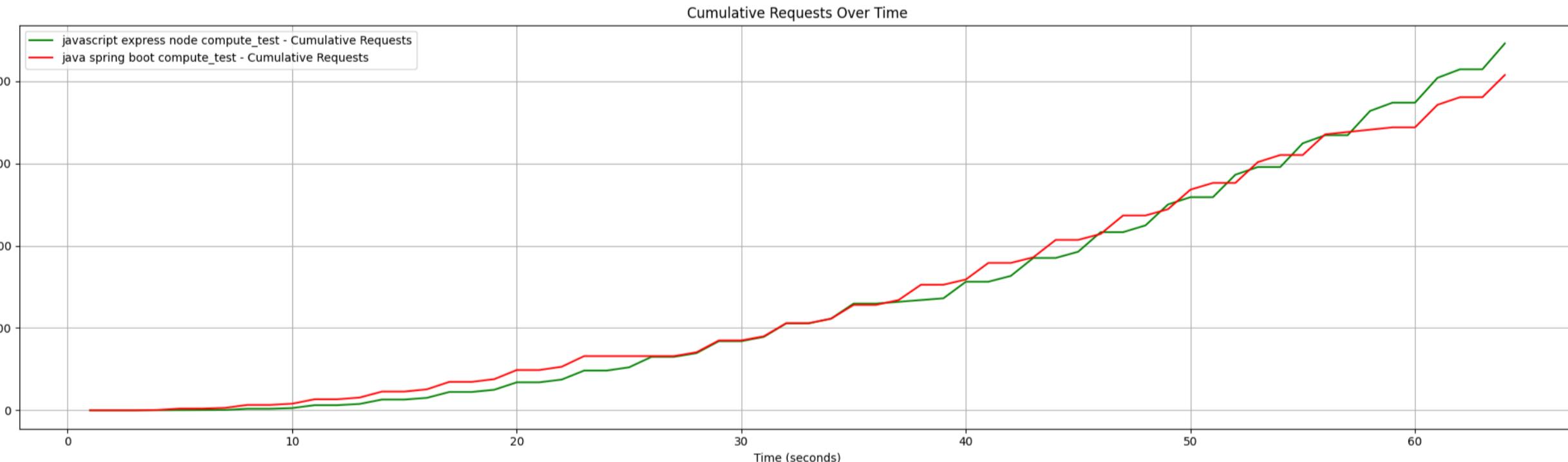
# Résultats



## **compute\_test**

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

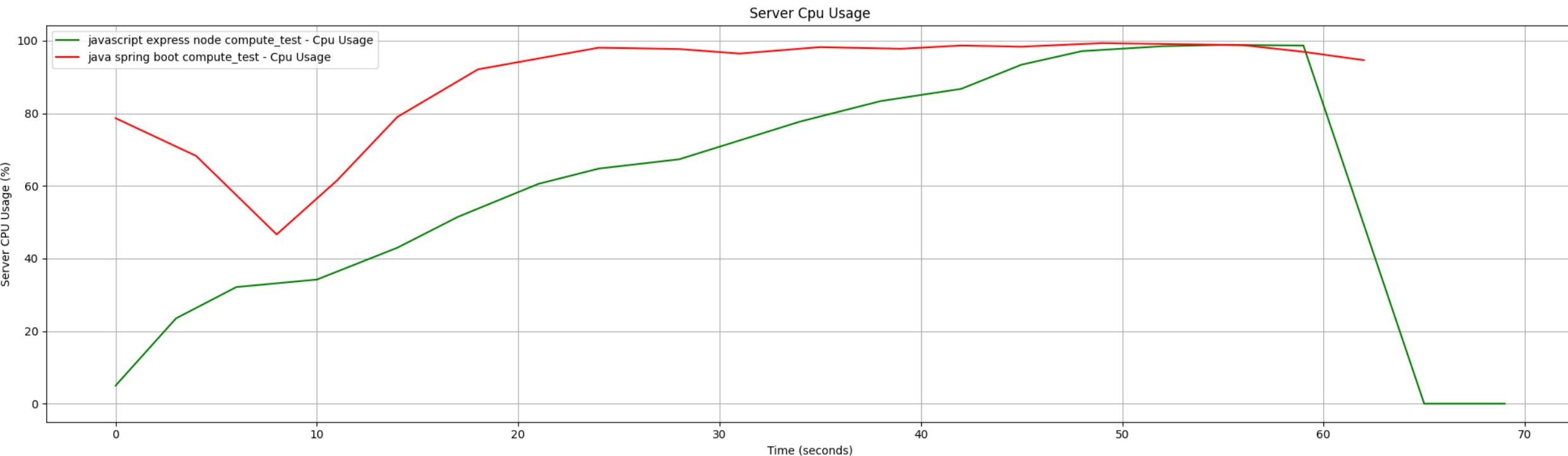
# Résultats



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

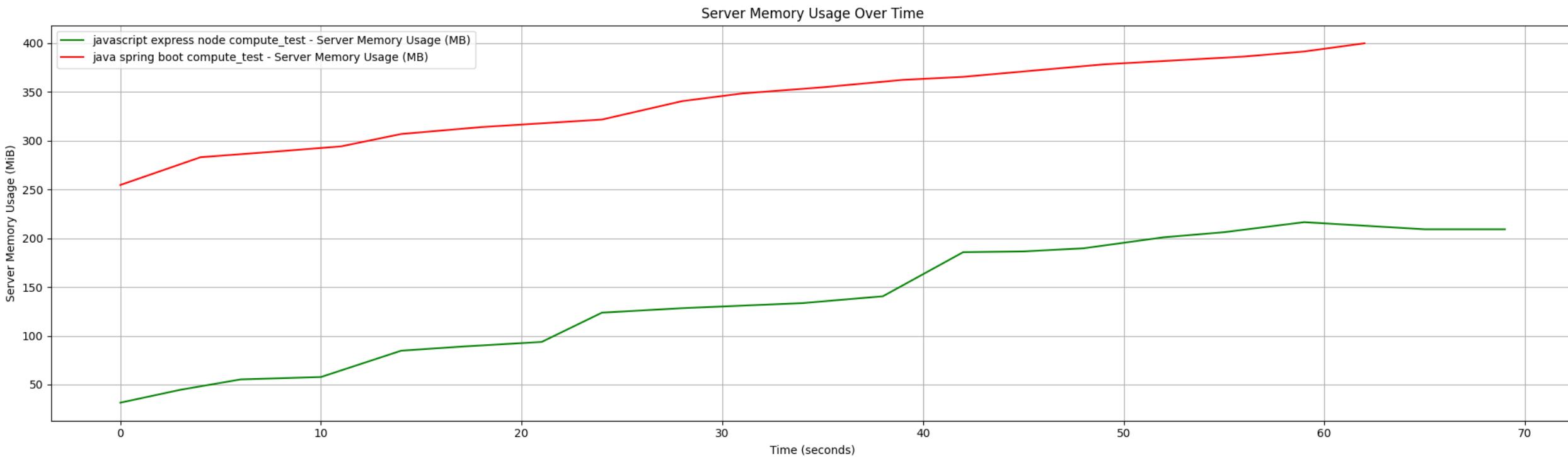
# Résultats



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

# Résultats



## compute\_test

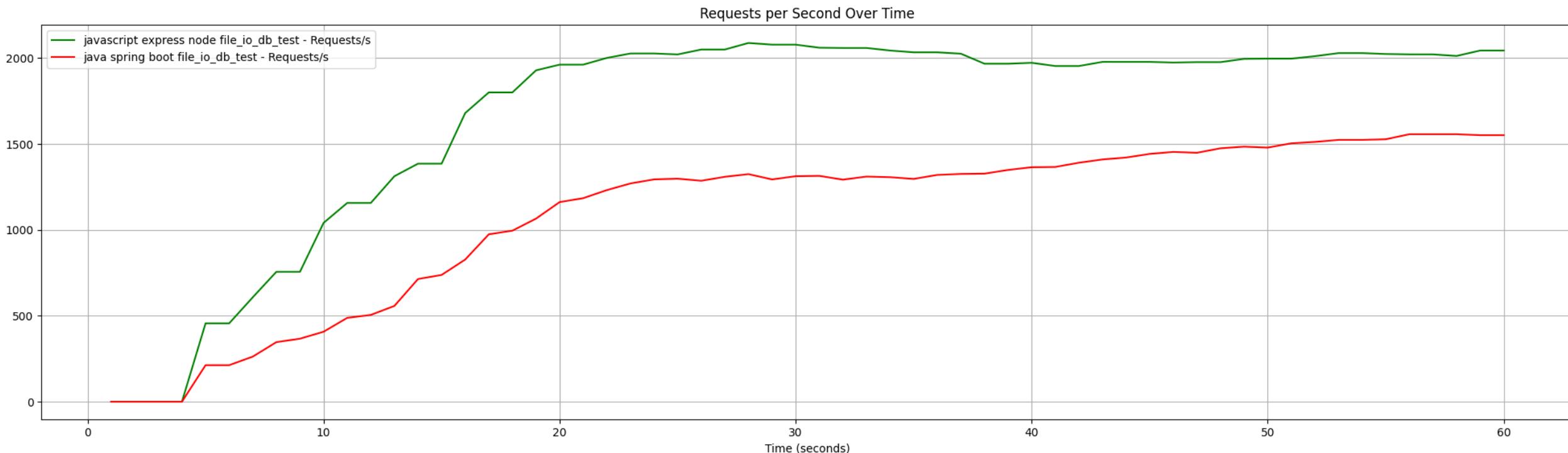
-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

# Résultats

## **file\_io\_db\_test**

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

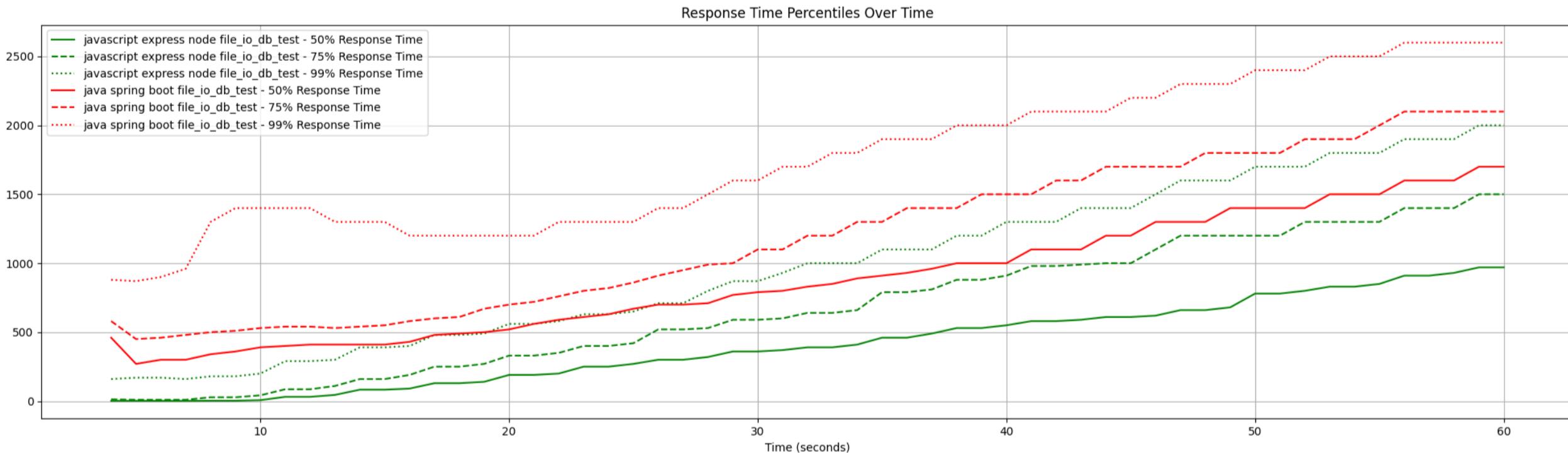
# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

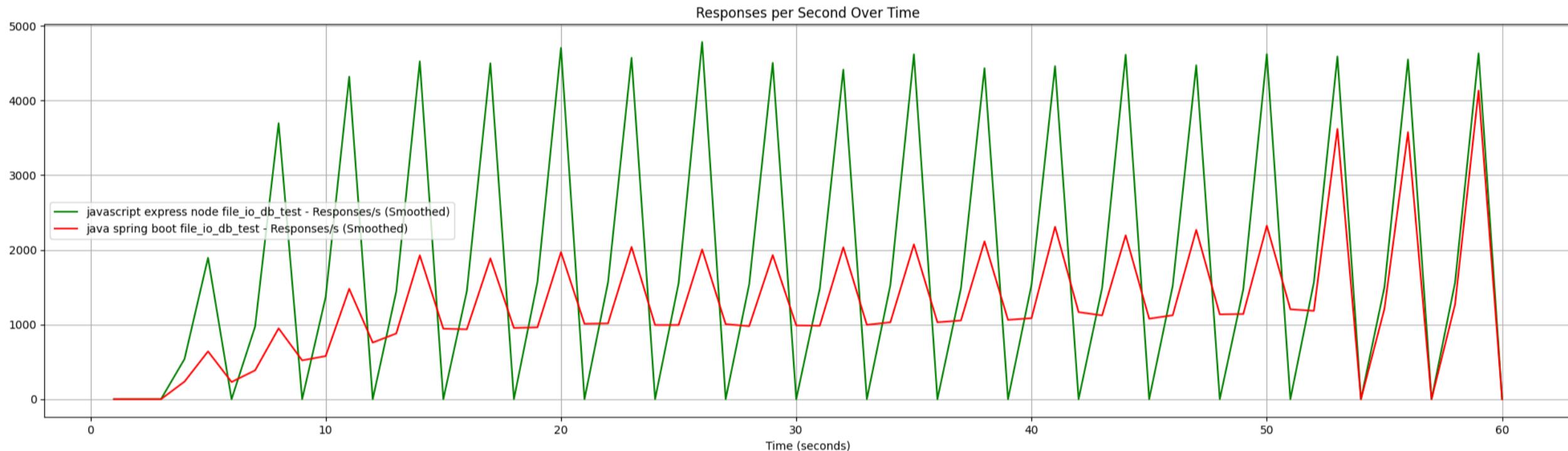
# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

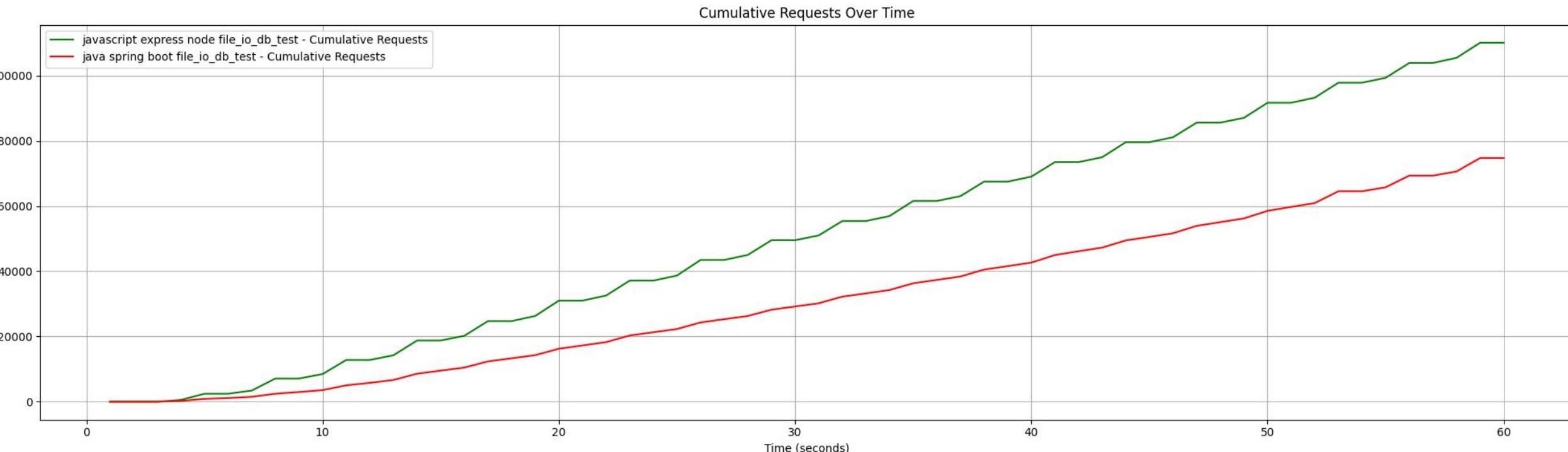
# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

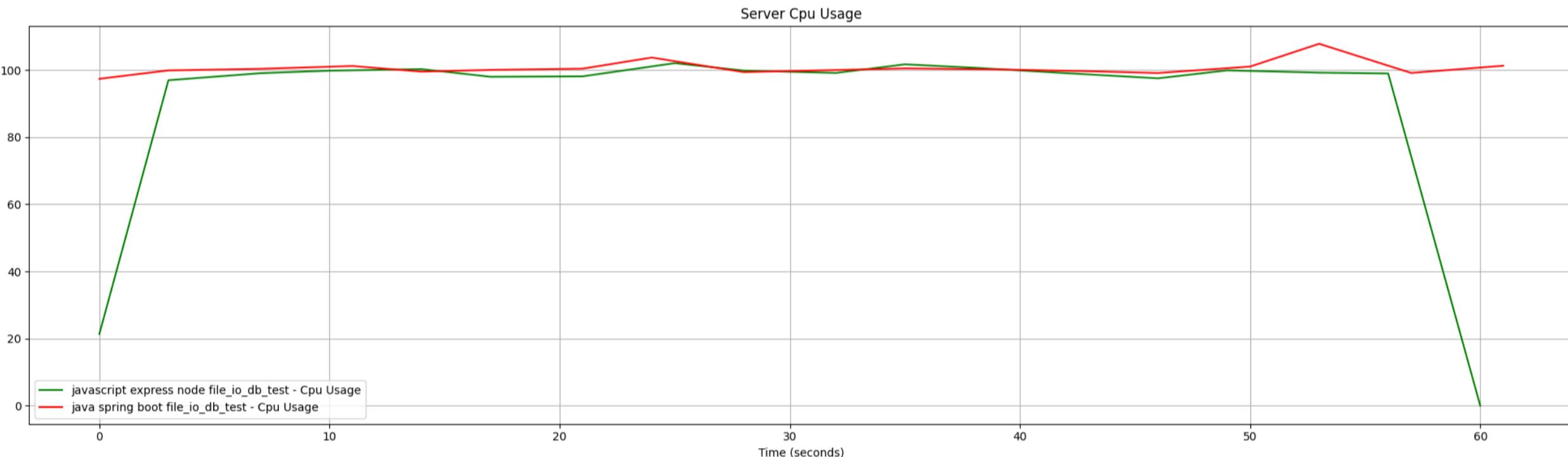
# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

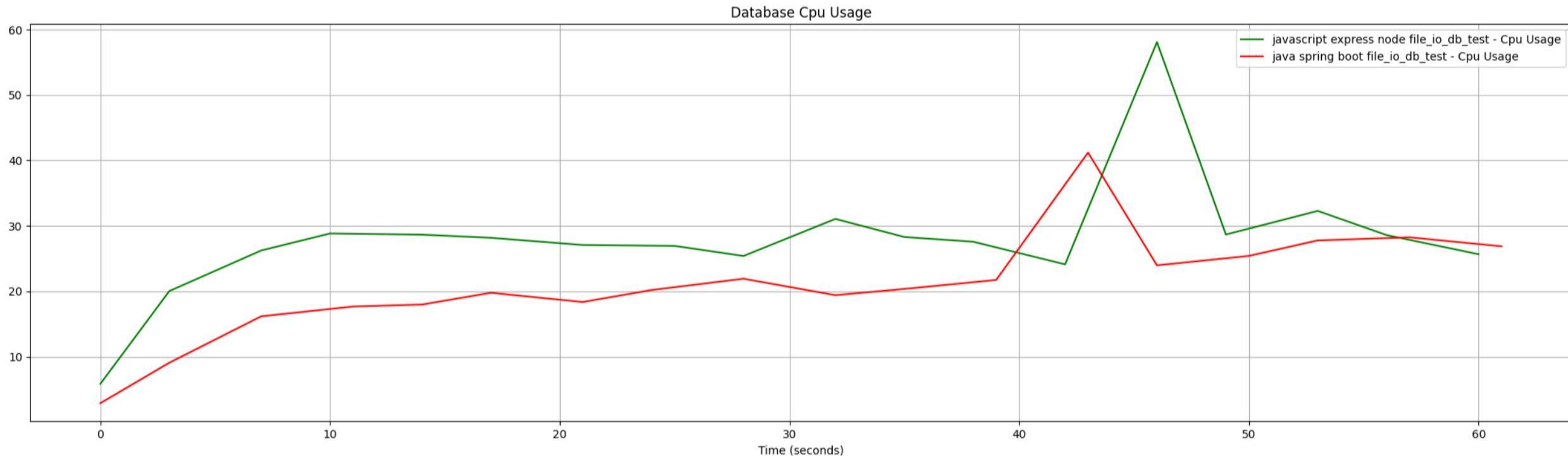
# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

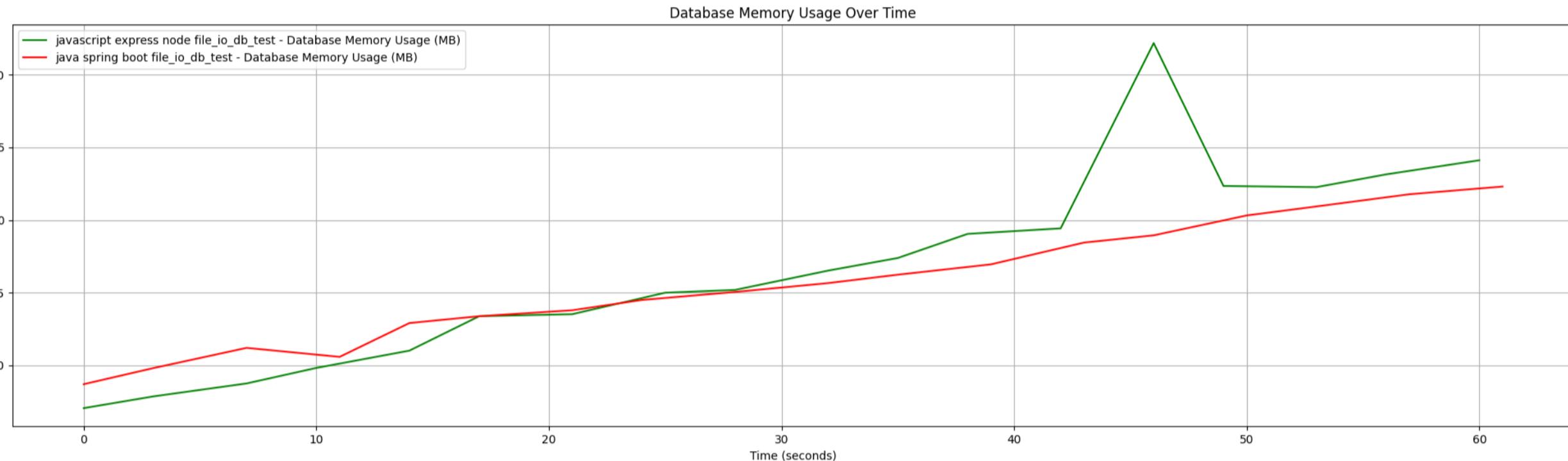
# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

# Résultats



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

# Résultats

## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

```
JS app.js
<> index.html
🖼 rl_icon.png
"># styles.css
```

Guide SSL Rocket League × +

localhost:8000/webpage/



COACHING ROCKET LEAGUE EXPRESS

## La Route vers le rang SSL

Découvre comment optimiser ton grind : routines mécaniques, travail de rotation et préparation mentale pour atteindre Supersonic Legend sans perdre des saisons entières.

[Lancer la session](#)

### Champions Field

#### SSL Grind · Focus Total

Routine du jour : 45 min de mech labs, 30 min de private match, 10 games ranked.

#### Maîtrise Mécanique

Speed flip, resets, wave dash recoveries : structure tes freestyle en blocs de 15 minutes et enregistre tes inputs pour corriger la caméra et la deadzone.

- Pack de custom trainings quotidiens
- Analyse des replays en slow-motion
- Checklist de constance aérienne

#### Rotations & Décisions

Apprends à lire la première touche adverse, à stabiliser la vitesse et à communiquer les boosts. L'objectif : zéro double commit au rank Grand Champion.

- Maps d'objectifs pour chaque rang
- Calls rapides en 3 mots max
- Template de review après scrum

#### Mindset & Tempo

Gestion du tilt, micro-pauses entre les séries et objectifs hebdomadaires réalistes pour grimper en SSL sans burnout.

- Protocoles anti-tilt en 90s
- Planning de scrum progressif
- Journal de bord post-match

### Journal SSL

Suis les retours des grinders : micro-ajustements de sensibilité, nouveaux packs de training et playlists recommandées pour rester en flow pendant les sessions.

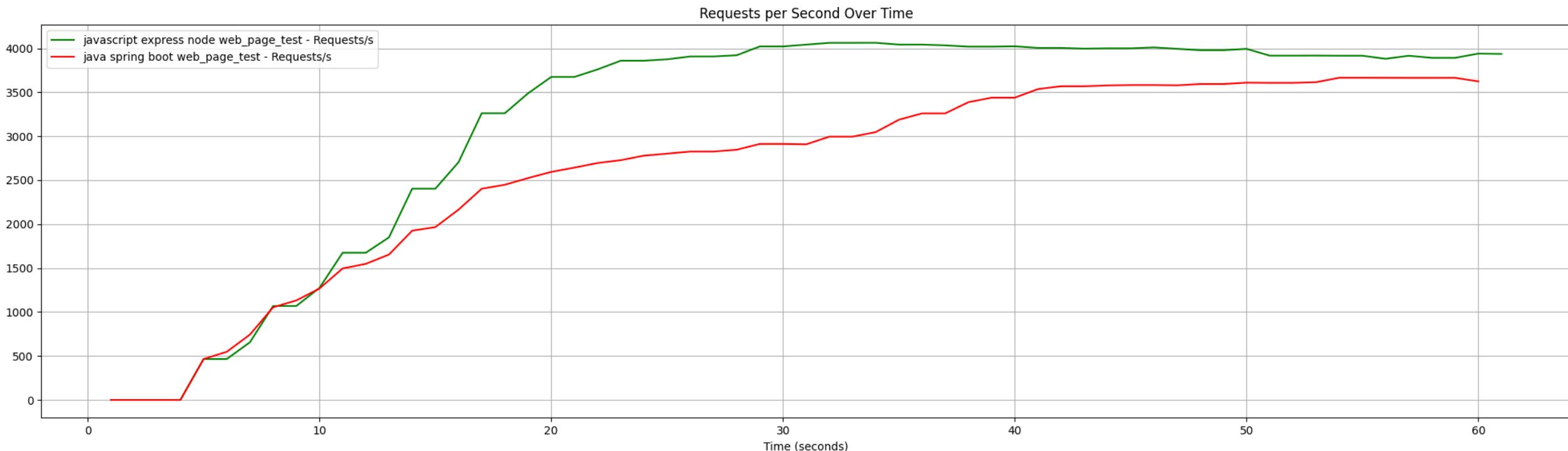
Dispatch #1Alban

Jouez avec YS94v2 pour vous faire carry en 2v2, il est chaud en ce moment.

Dispatch #2Romain

Partir en aérienne ? Je connais pas. Focus sur les 50/50 au sol et les rotations basiques.

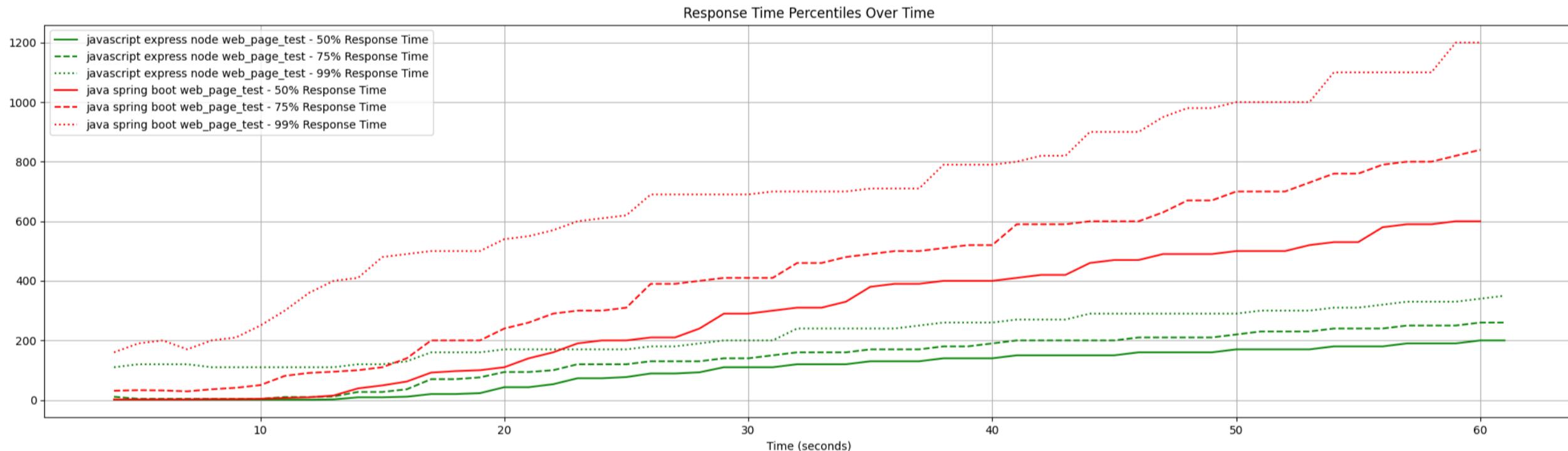
# Résultats



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

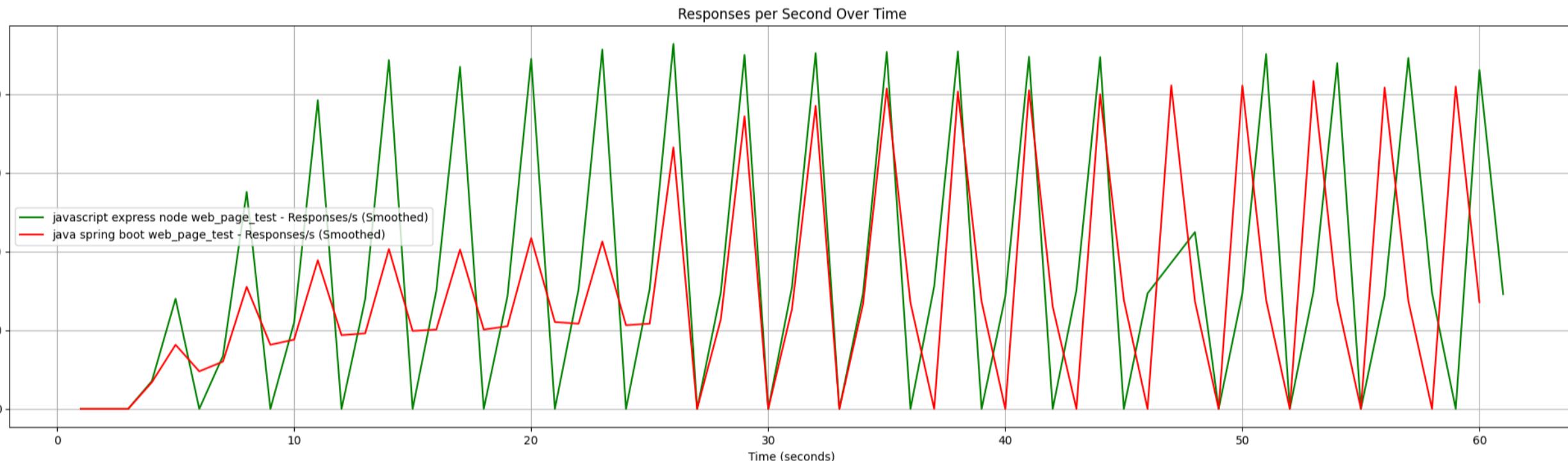
# Résultats



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

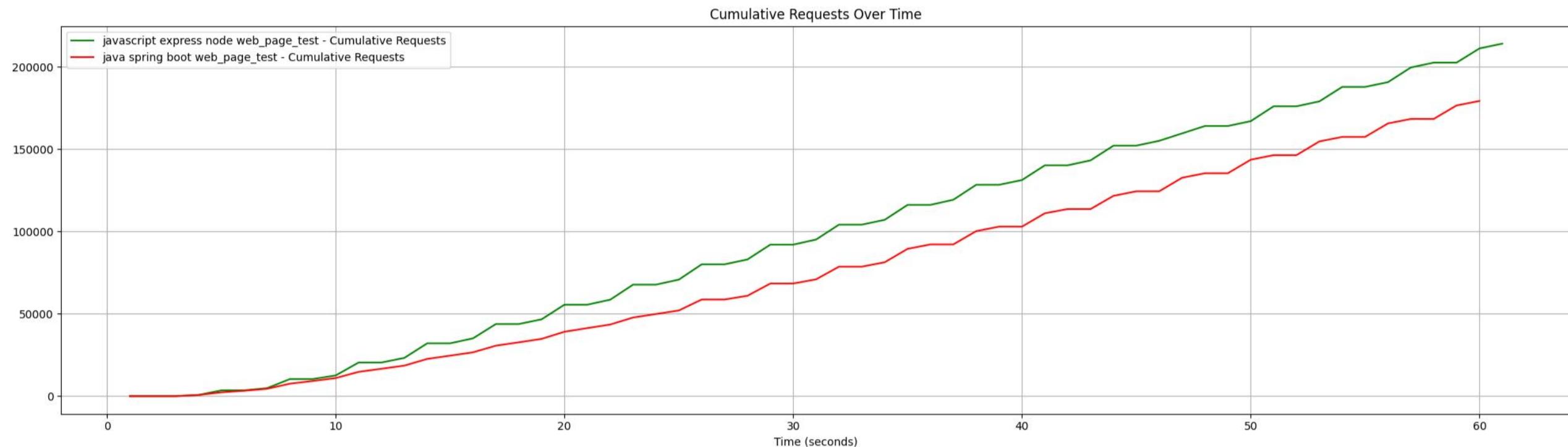
# Résultats



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

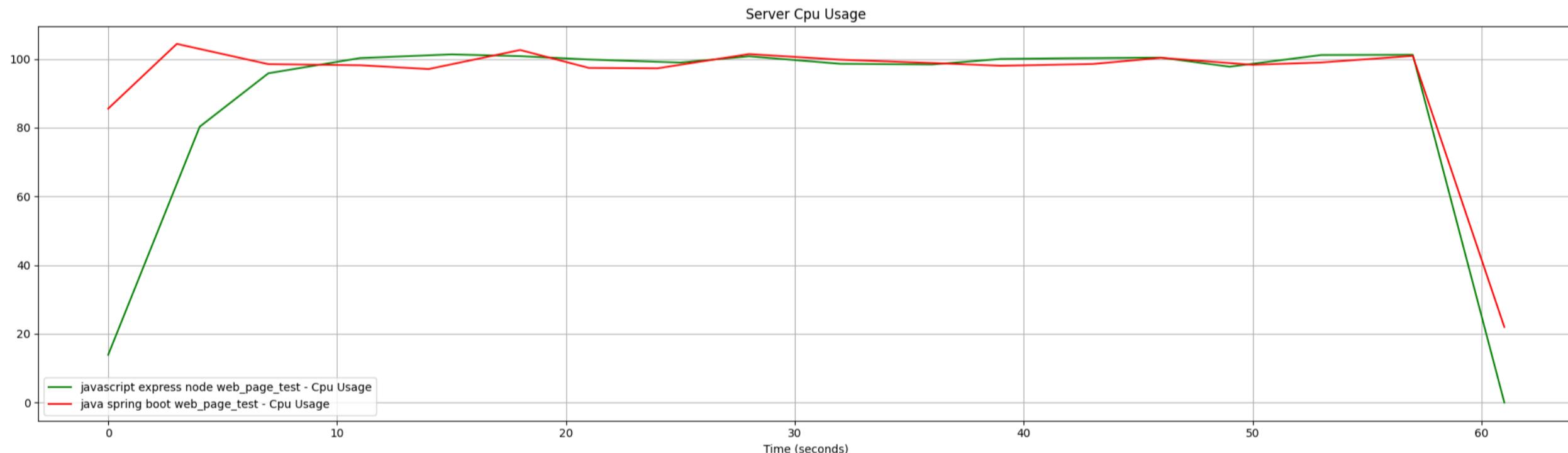
# Résultats



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

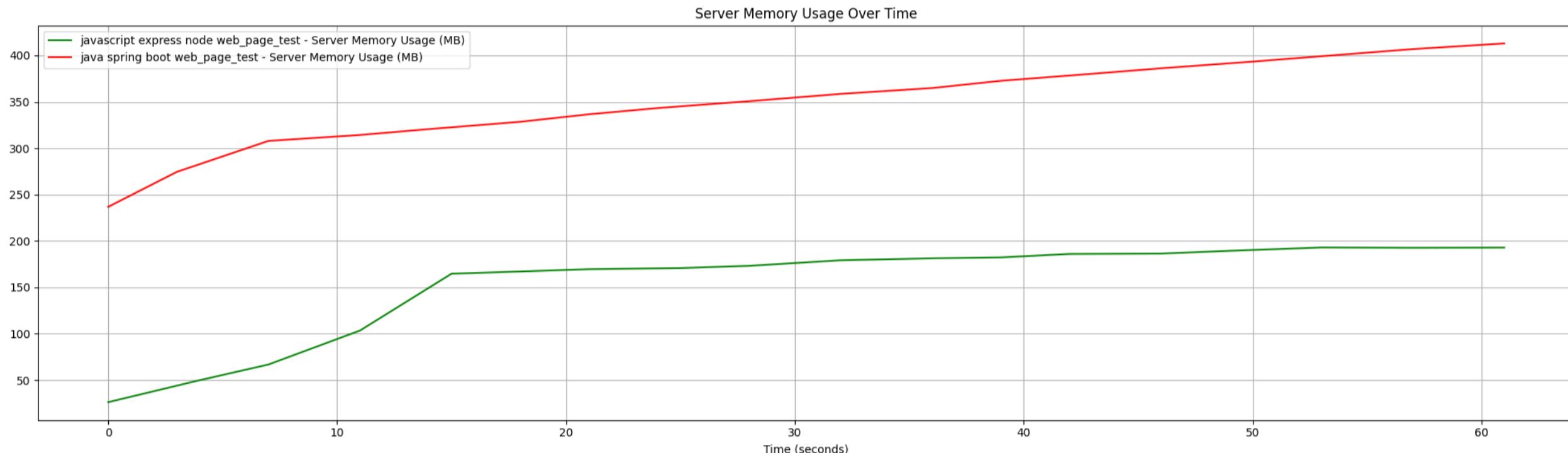
# Résultats



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

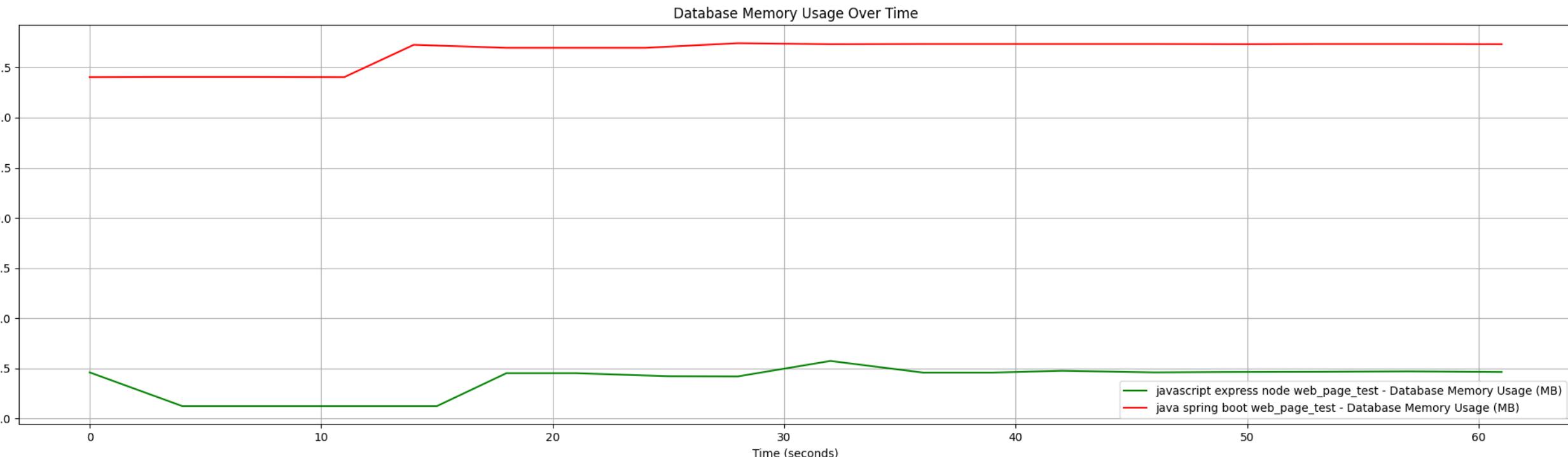
# Résultats



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

# Résultats

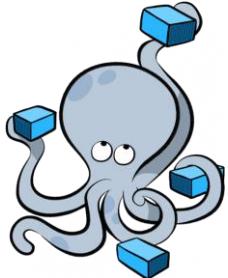


## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

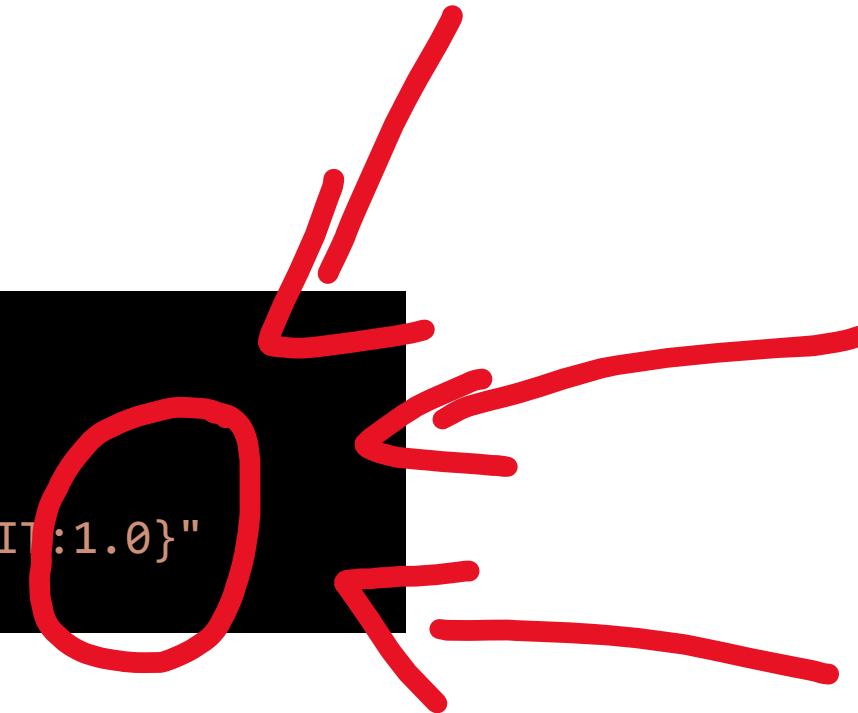
**Sympa les résultats, mais...**

# Sympa les résultats, mais...



«docker-compose.yml»

```
benchmark:  
# ...  
deploy:  
  resources:  
    limits:  
      cpus: "${SERVICE_CPU_CORE_LIMIT}:1.0"
```



**Nous étions bridés à un seul cœur de processeur**

# Sympa les résultats, mais on va augmenter les cœurs

AMD Ryzen 7 7700

```
SERVICE_CPU_CORE_LIMIT=16.0
```

【Number of cores】 : 8 Core	【Process technology】 : 5nm
<b>【Threads】 : 16 Thread</b>	【TDP power】 : 65W
【CUP frequency】 : 3.8-5.3GHz	【Architecture】 : Zen 4
【RAM】 : DDR5 5200MT/s	【Game cache】 : 40MB

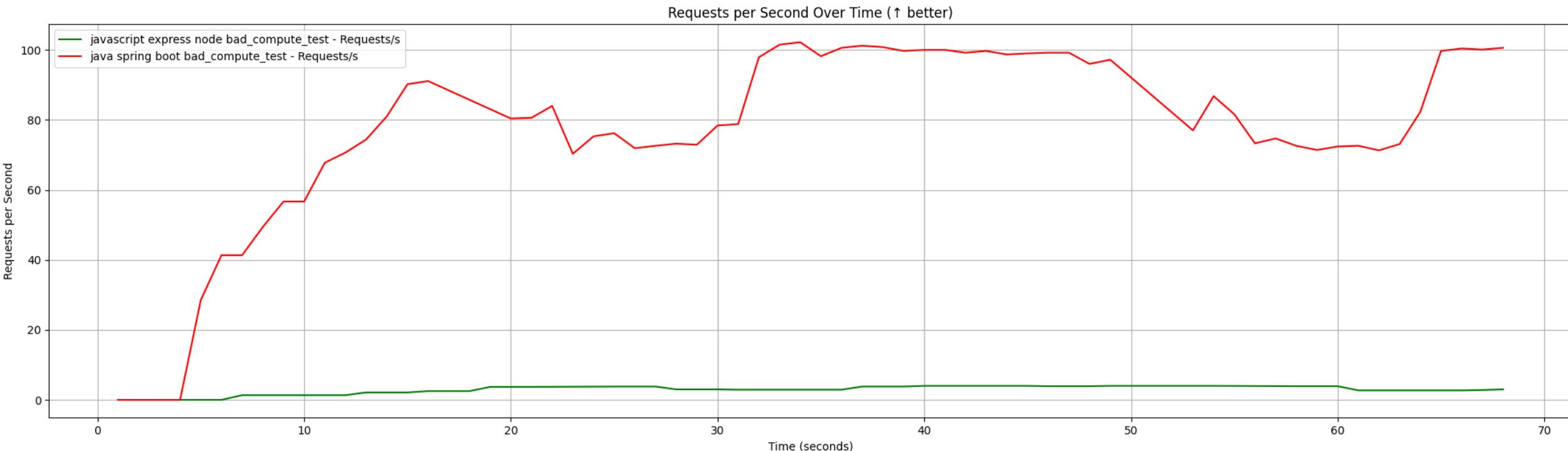


# Résultats (16 cœurs)

## **bad\_compute\_test**

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

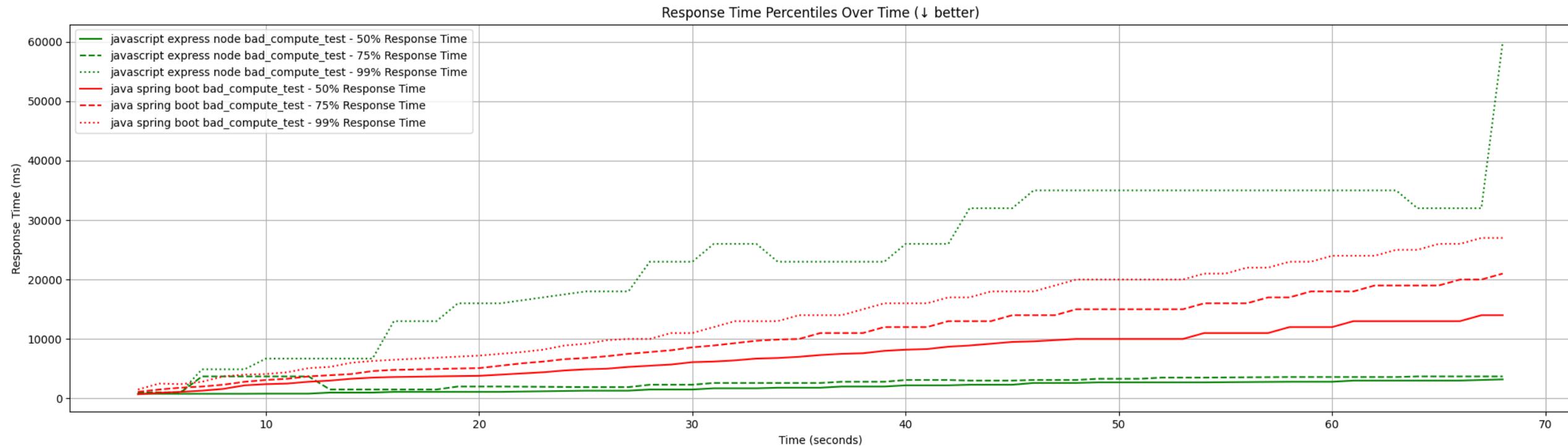
# Résultats (16 cœurs)



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

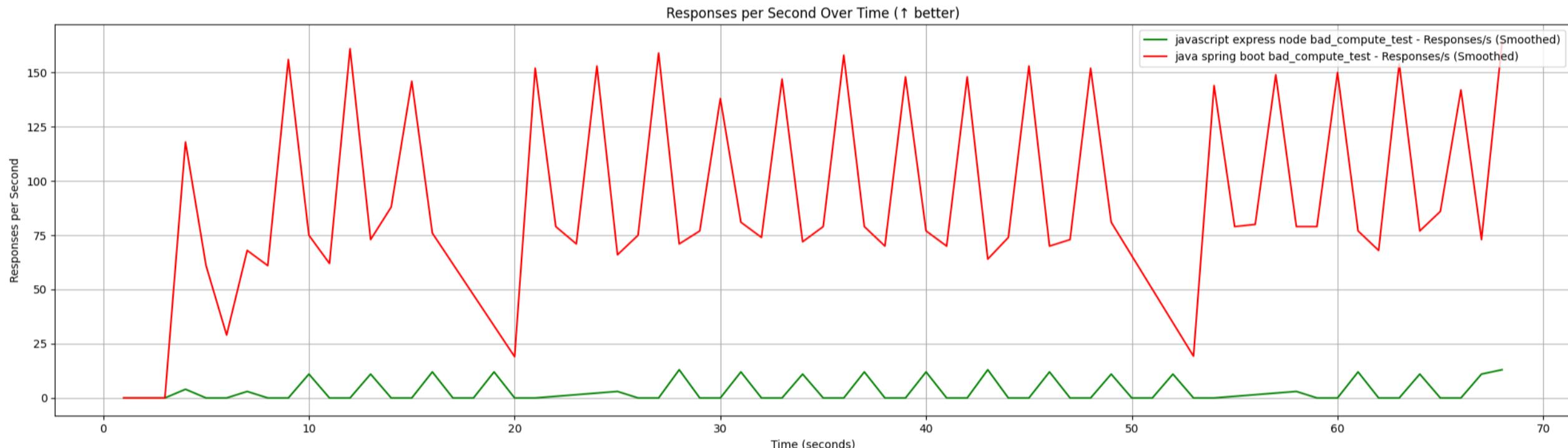
# Résultats (16 cœurs)



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

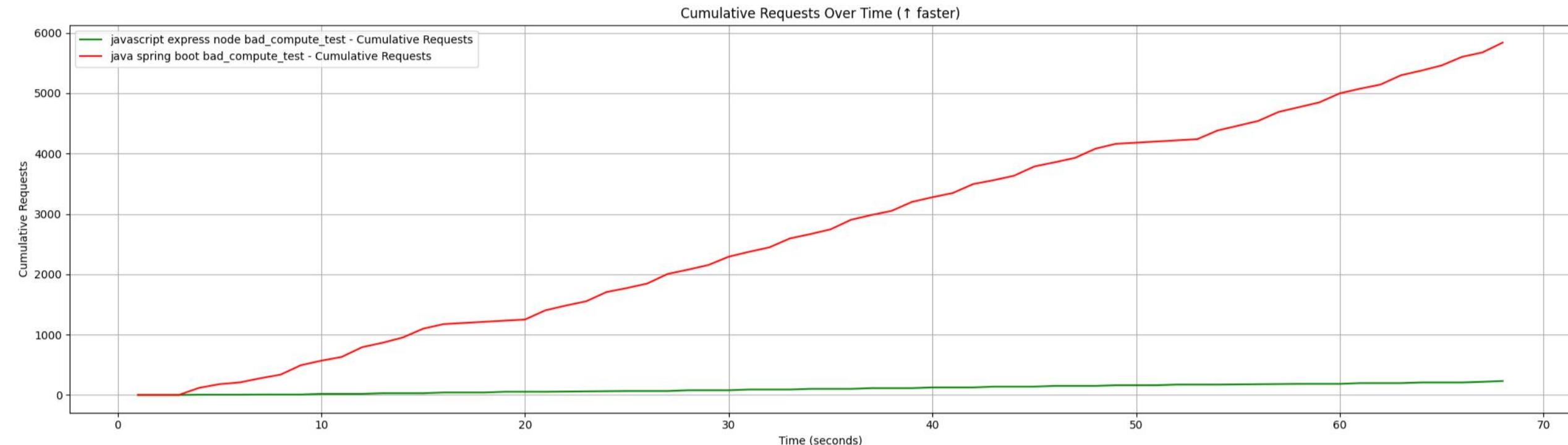
# Résultats (16 cœurs)



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

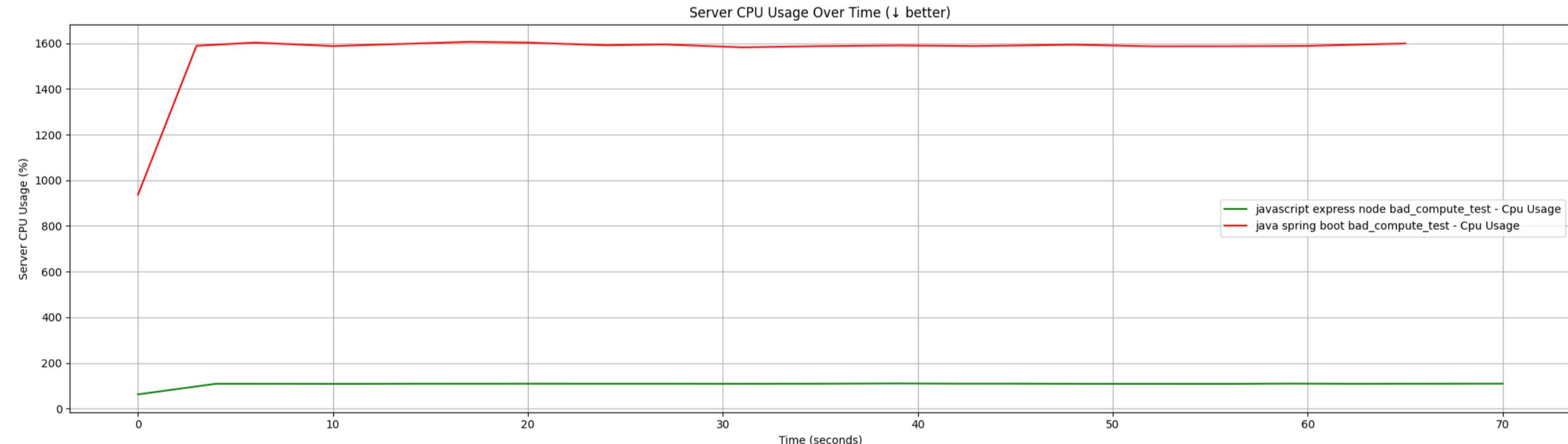
# Résultats (16 cœurs)



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

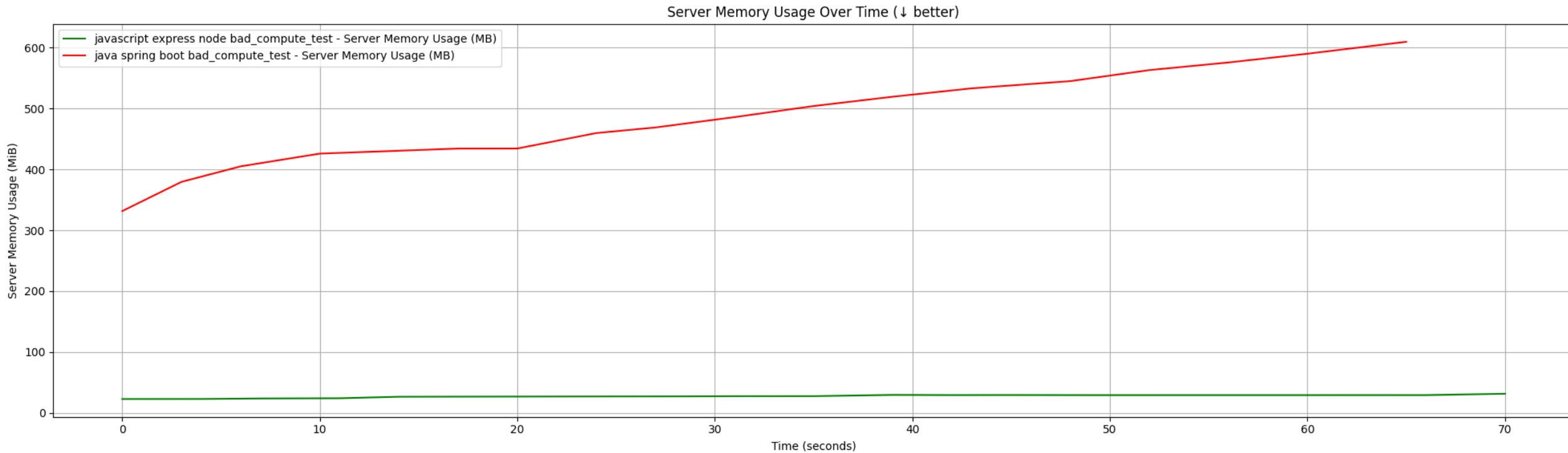
# Résultats (16 cœurs)



## bad\_compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

# Résultats (16 cœurs)



## bad\_compute\_test

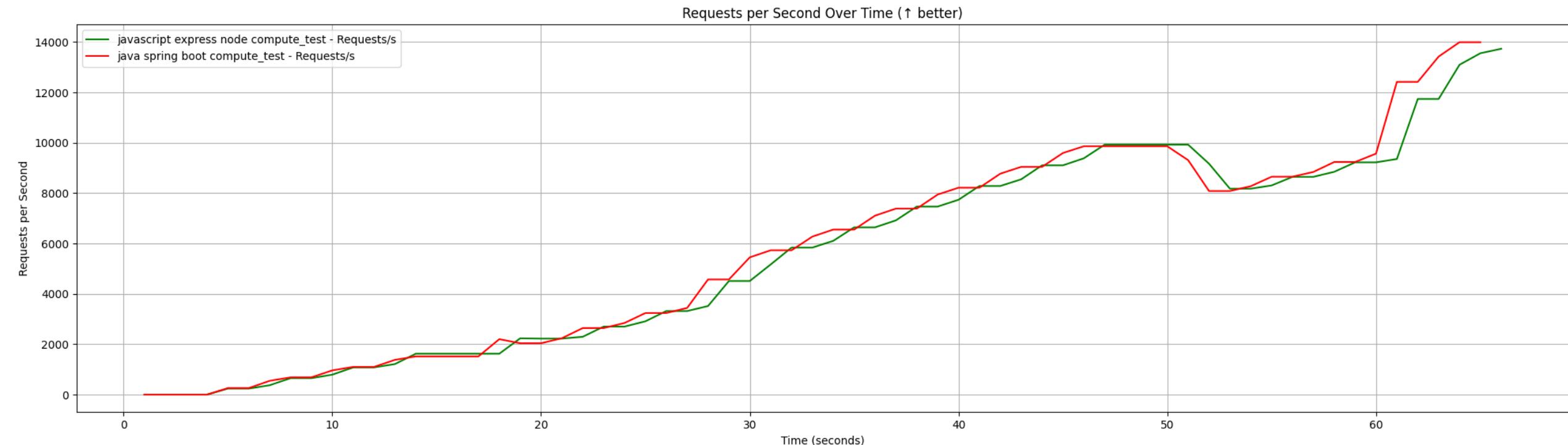
-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 38 **RECURSIF** et renvoie un OK(200)

# Résultats (16 cœurs)

## **compute\_test**

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

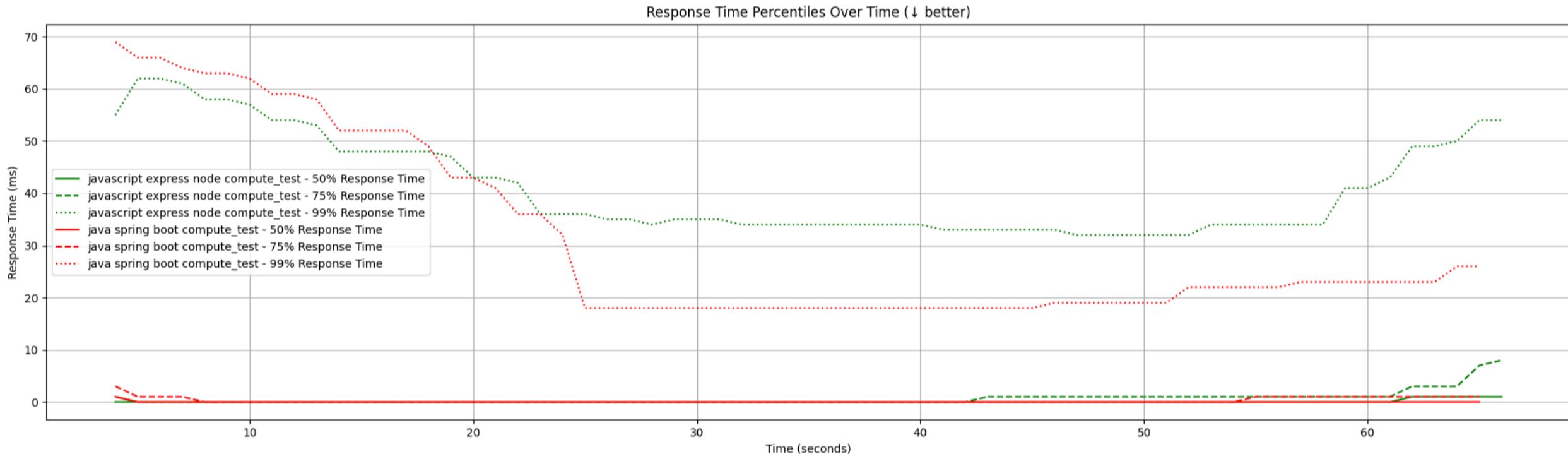
# Résultats (16 cœurs)



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

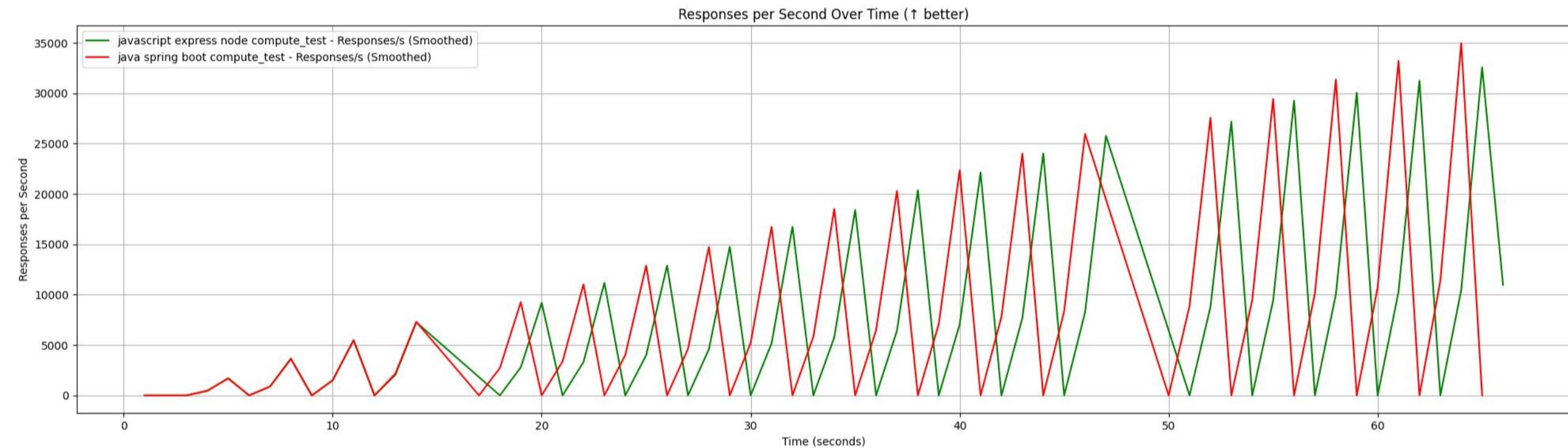
# Résultats (16 cœurs)



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

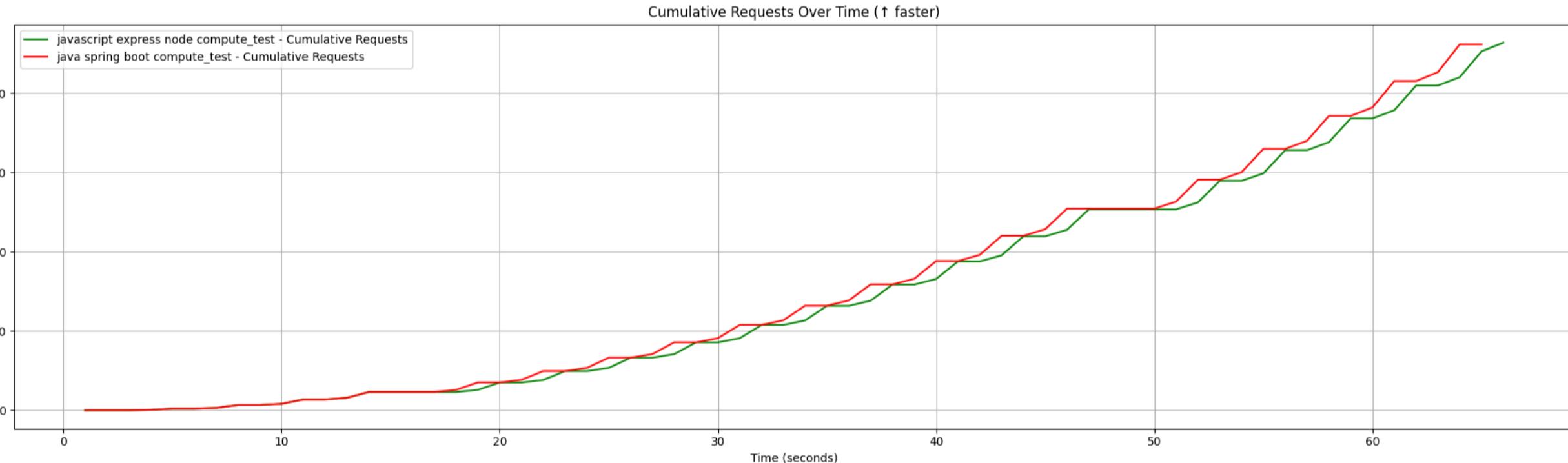
# Résultats (16 cœurs)



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

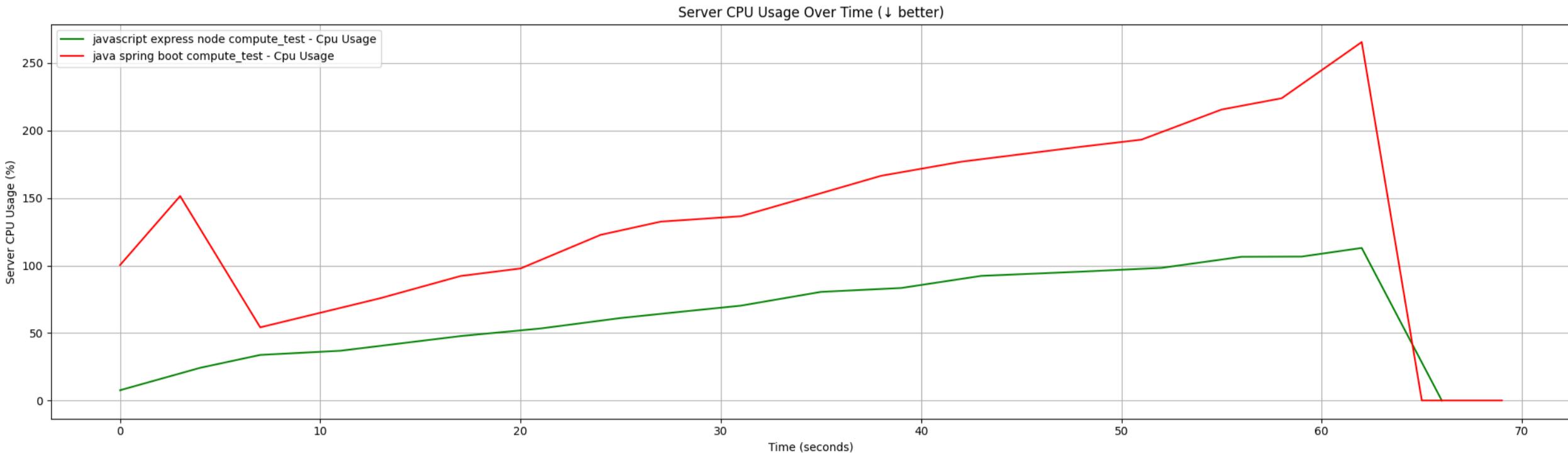
# Résultats (16 cœurs)



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

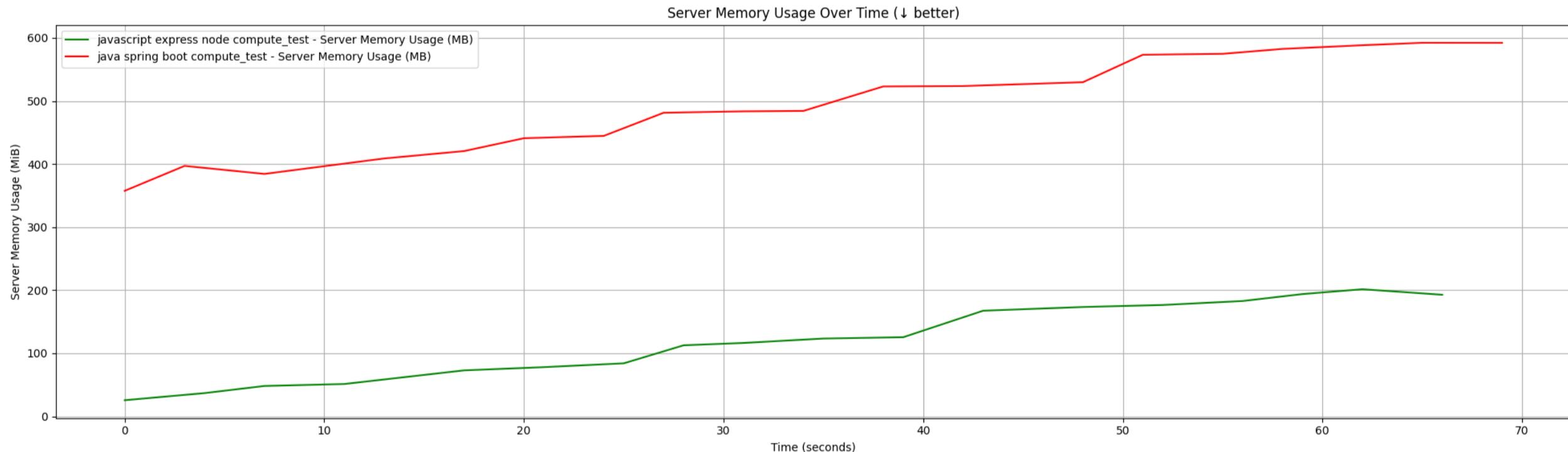
# Résultats (16 cœurs)



## compute\_test

-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

# Résultats (16 cœurs)



## compute\_test

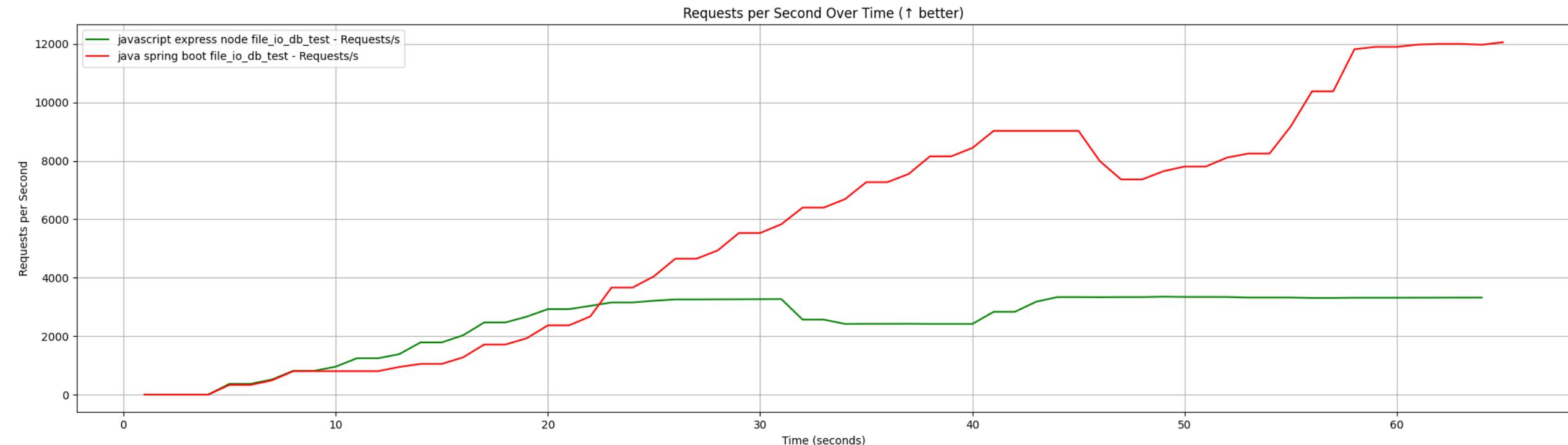
-> GET(«/compute/fibonacci/») : Calcule un Fibonacci de 42 et renvoie un OK(200)

# Résultats (16 cœurs)

## **file\_io\_db\_test**

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

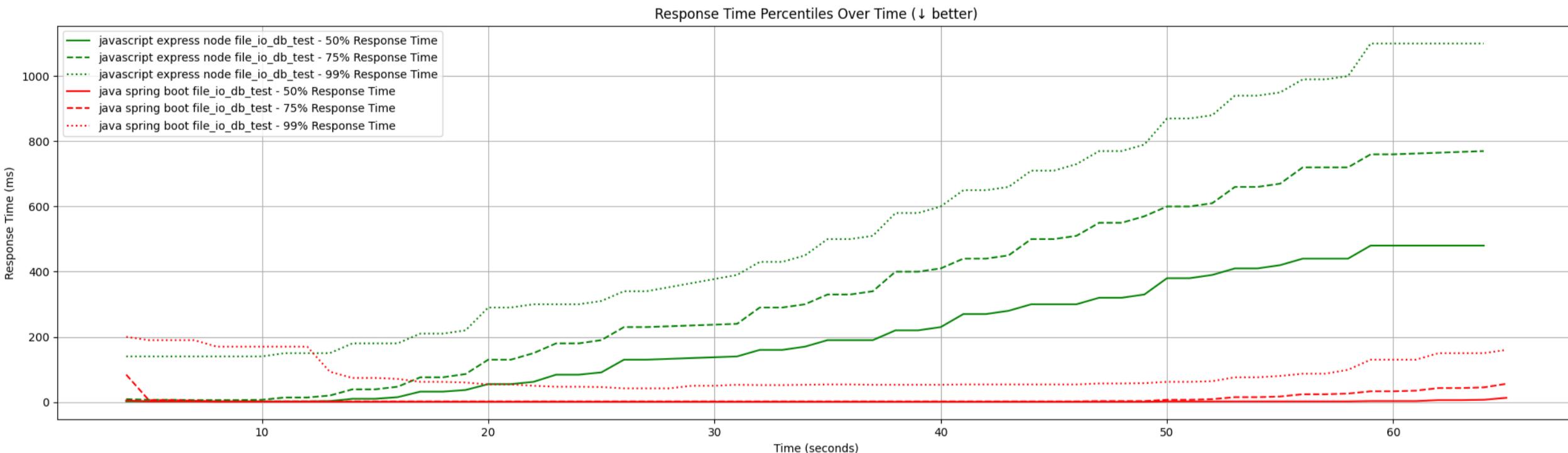
# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

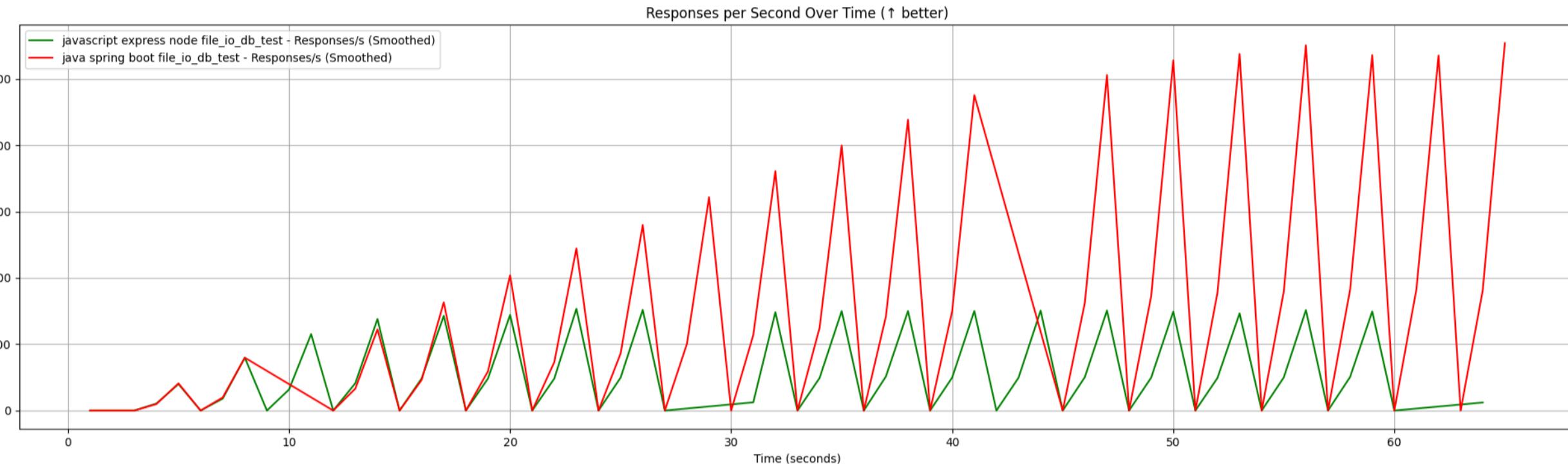
# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

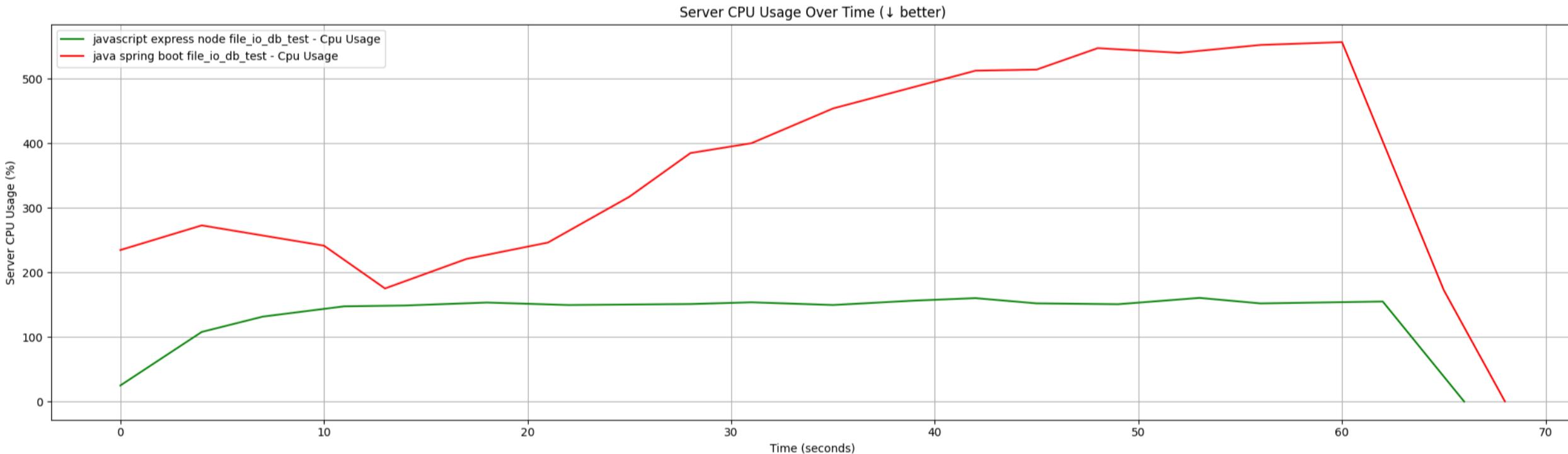
# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

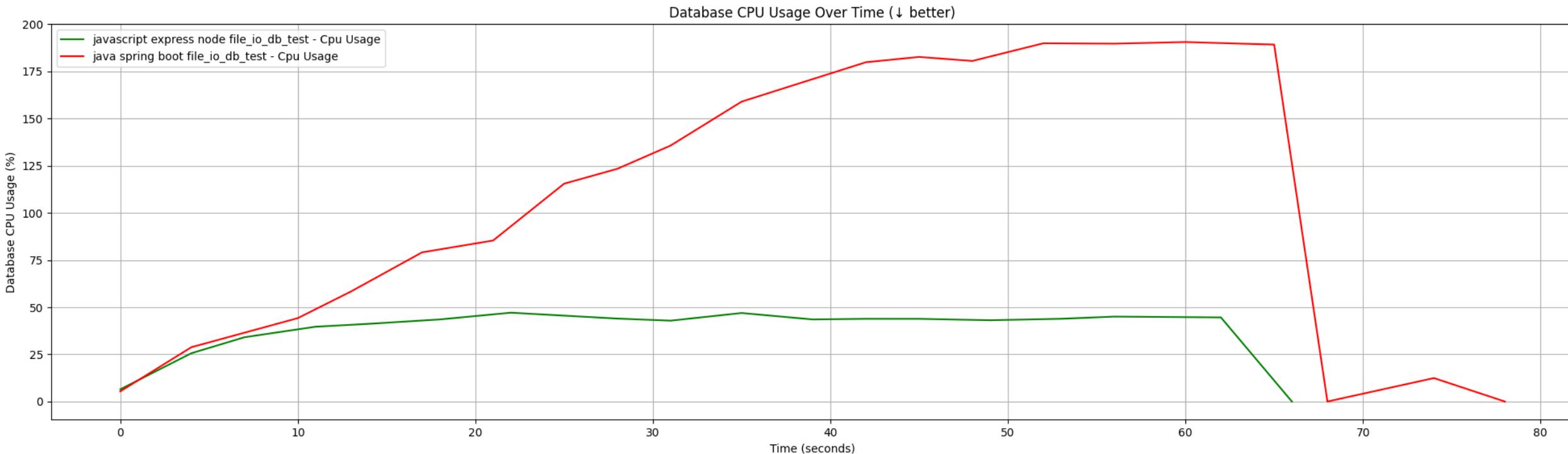
# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

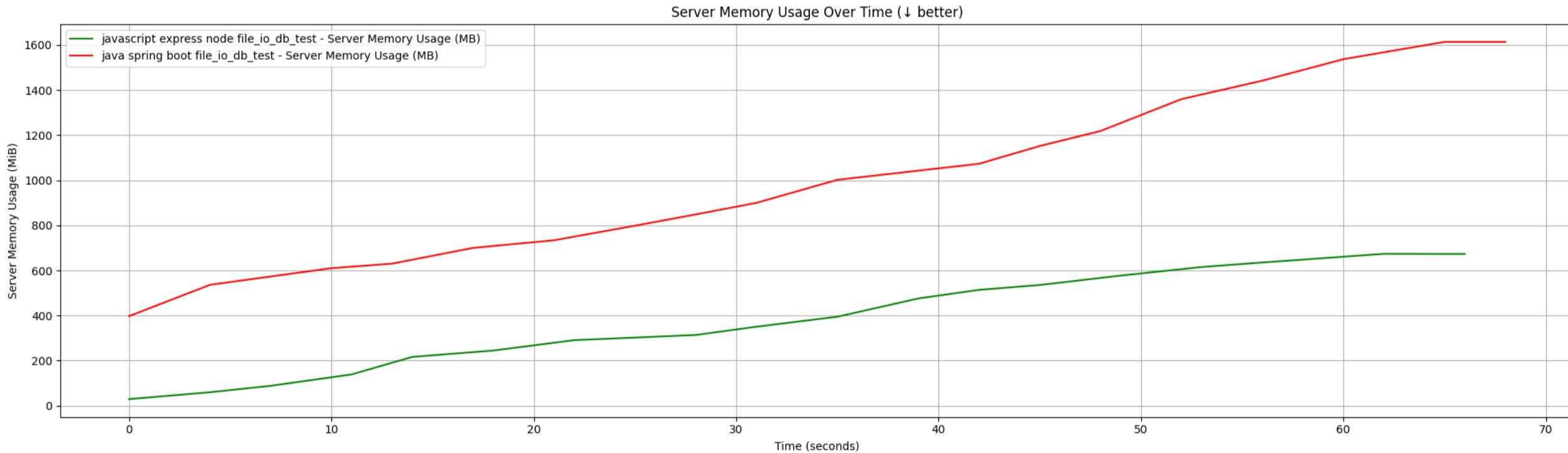
# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

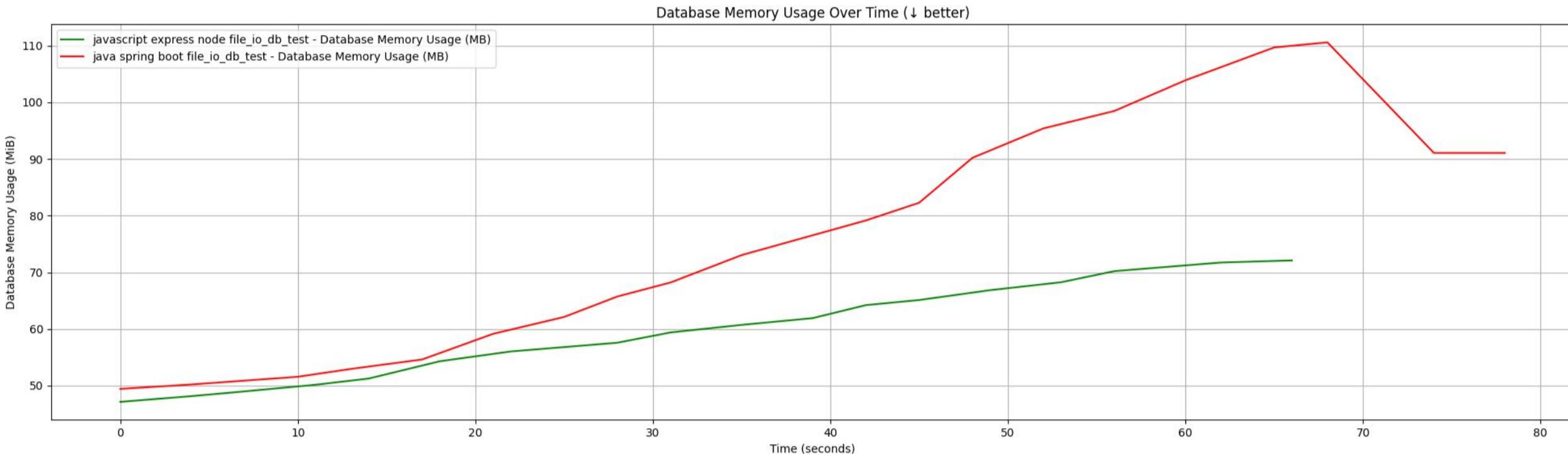
# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

# Résultats (16 cœurs)



## file\_io\_db\_test

- > POST(«/notes/file\_io/») : Crée un fichier et écrit la note dedans, puis la save en BDD, et lit le fichier
- > GET(«/notes/file\_io/latest/») : Cherche en BDD le path de la dernière note, lit le contenu de ce fichier et renvoie un OK(200)

# Résultats (16 cœurs)

## **web\_page\_test**

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

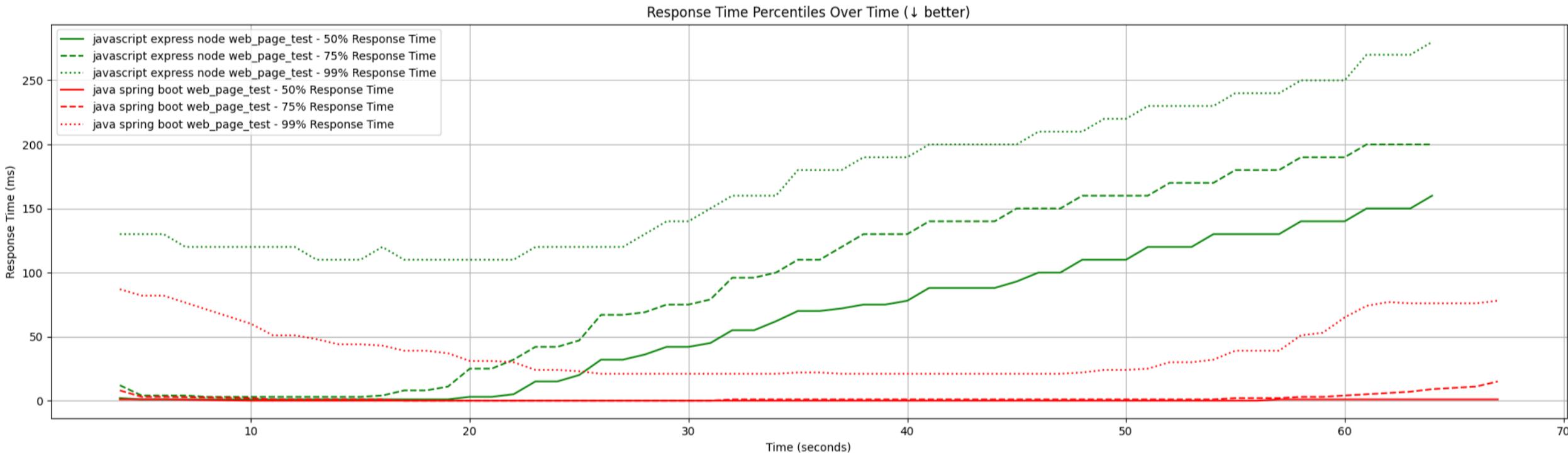
# Résultats (16 cœurs)



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

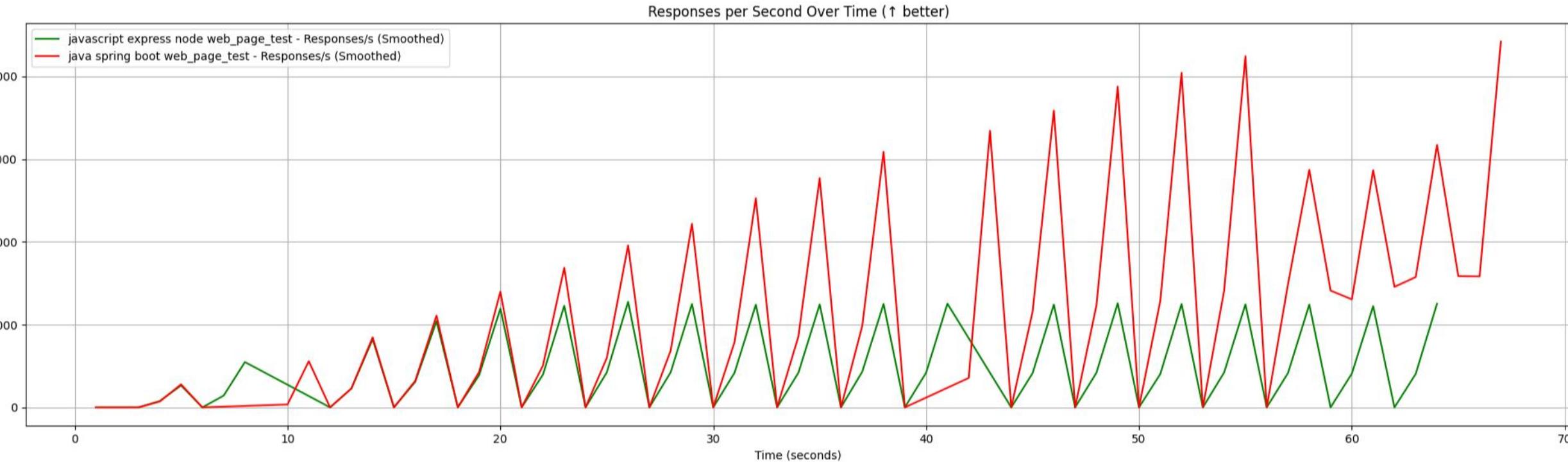
# Résultats (16 cœurs)



## web\_page\_test

→ GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

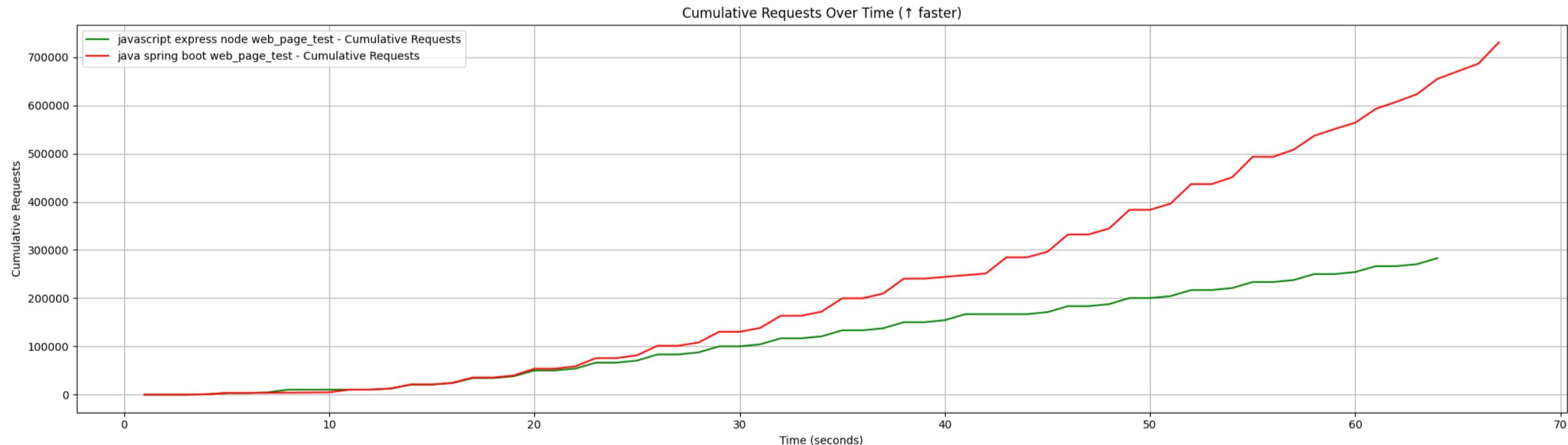
# Résultats (16 cœurs)



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

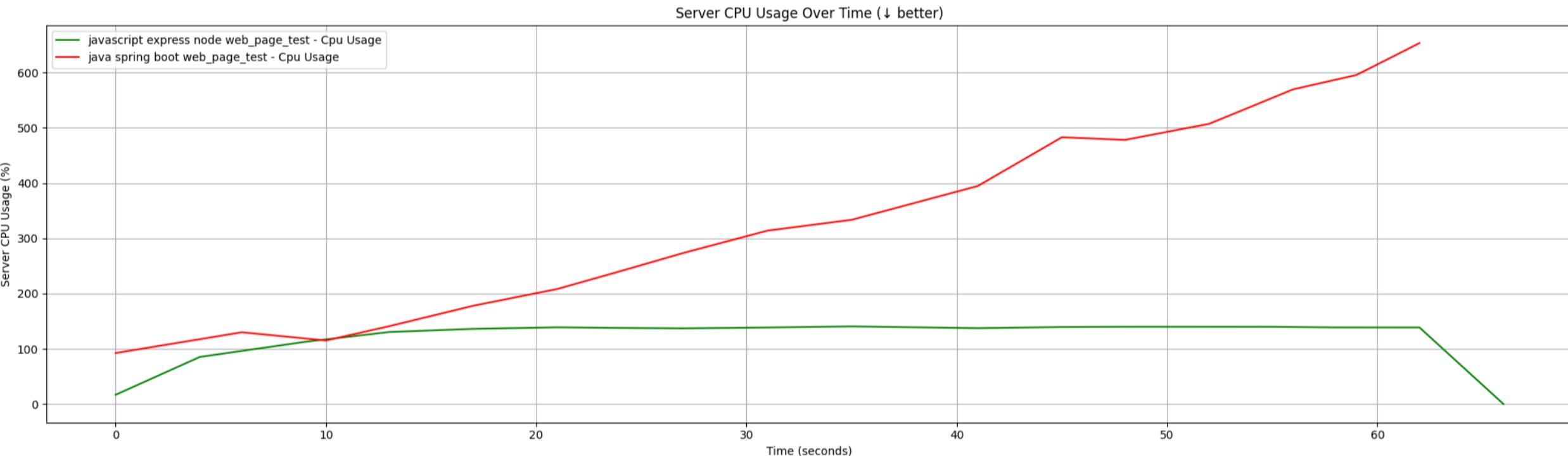
# Résultats (16 cœurs)



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

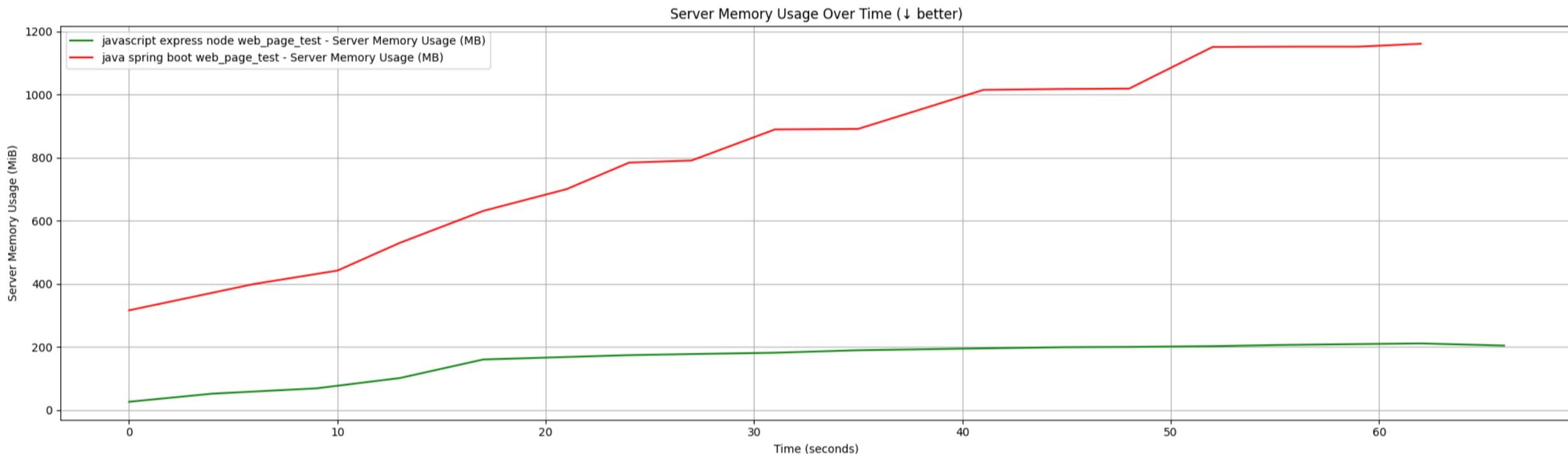
# Résultats (16 cœurs)



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

# Résultats (16 cœurs)



## web\_page\_test

-> GET(«/no\_db\_file\_io/») : Renvoie une page HTML statique avec ces assets (CSS, JS)

# Conclusion



Serveur simple / CPU bas de gamme

Serveur complexe (règles métier) / CPU milieu-haut de gamme



# Bibliographie

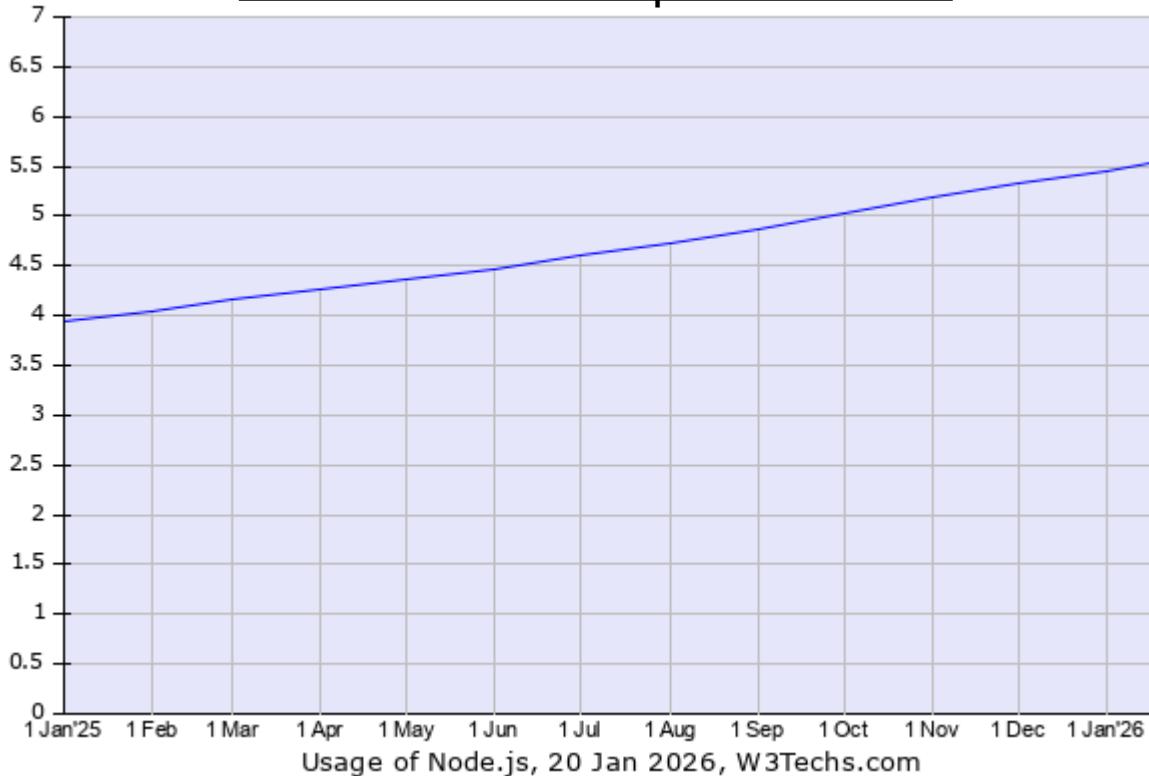
- [https://www.w3schools.com/nodejs/nodejs\\_architecture.asp](https://www.w3schools.com/nodejs/nodejs_architecture.asp) : Architecture NodeJs
- <https://www.axopen.com/technos/versus/spring-boot-vs-nodejs/> : Comparaison de Spring Boot et Node
- <https://kinsta.com/fr/blog/qu-est-ce-que-node-js/> : Explication de Node
- <https://jmeter.apache.org/> : Qu'est ce que JMeter ?
- [https://fr.wikipedia.org/wiki/Gatling\\_\(logiciel\)](https://fr.wikipedia.org/wiki/Gatling_(logiciel)) : Qu'est-ce que Gatling ?
- <https://github.com/abdelaziz-mahdy/backend-benchmark> : L'outil de benchmark utilisé
- <https://javascriptwtf.com/> : Bizarries Javascript
- <https://www.sfeir.dev/back/il-etait-une-fois-spring-boot> : Explication de Spring
- <https://www.sfeir.dev/back/how-to-trouver-ses-starters-springboot/> : Explication des starters
- <https://www.ionos.fr/digitalguide/sites-internet/developpement-web/framework-spring-pour-votre-infrastructure-java/> : Conception et fonctionnement de spring
- <https://makina-corpus.com/front-end/introduction-nodejs> : Explication NodeJs

# Bibliographie

- <https://spring.io/> : Site officiel Spring
- <https://www.youtube.com/watch?v=mzX6DX1S7Fc> : Vidéo d'explication NodeJs
- <https://datascientest.com/framework-vs-library> : Définition framework et librairie
- <https://kinsta.com/blog/what-is-node-js/> : Explication de NodeJs
- <https://medium.com/@cindanojonathan.fr/5-points-de-comparaison-entre-spring-boot-et-node-js-7895a3f1b207> : Comparaison de Spring Boot et Node
- <https://www.youtube.com/watch?v=k7sn7dBh4eQ> : Vidéo explication NodeJs
- <https://youtu.be/ogNZf6BULac?si=aNJtTqCaetGJt-le> : Vidéo expliquant le déploiement de NodeJs en prod
- <https://youtu.be/yjJALFHAb-o?si=6VaU0etdgotYTOH6> : Intégration de Tomcat
- <https://youtu.be/w0fdzTbXitg?si=I0sj0eX18GcDHwro> : Explication de Java EE

# Annexes

## Taille du marché pour NodeJs



## Applications :

- Twitter
- Spotify
- eBay
- Reddit
- LinkedIn
- Godaddy
- Netflix
- Paypal

Source : <https://w3techs.com/technologies/details/ws-nodejs>

# Annexes

Le **typage dynamique** signifie qu'un compilateur ou un interpréteur attribue un type à toutes les variables lors de l'exécution

Un **langage à typage statique** est un langage avec lequel les types des variables sont connus lors de la compilation et doivent être spécifiés expressément par le programmeur. Le type donné à la compilation ne doit pas changer pendant l'exécution.

# Annexes

**Framework :** Il s'agit d'une structure logicielle qui fournit un **ensemble de règles, de conventions, et de composants prédéfinis** pour faciliter le développement d'applications complètes. Il offre un cadre standardisé qui guide le développeur dans l'organisation du code et l'architecture de l'application, en définissant la manière dont les composants interagissent entre eux.

**Librairie :** Une librairie est une **collection de fonctions, de classes, ou de modules pré-écrits** que les développeurs peuvent utiliser pour accomplir des tâches spécifiques dans leurs projets. Contrairement à un framework, une bibliothèque n'impose pas de structure ou de flux particulier au développement d'une application. Elle fournit simplement des outils que le développeur peut appeler à sa convenance pour accomplir des tâches spécifiques, telles que la manipulation de données, l'interaction avec une API, ou la gestion de l'interface utilisateur.

# Annexes

Pour **SpringBoot**, par défaut, on a un **thread request pool** de **200** :

- **1 cœur CPU** : Ce seul cœur CPU va se partager les 200 threads
- **16 cœurs CPU** : Les 200 threads seront étalés sur 16 cœurs, et on débloque donc 16 réelles exécutions parallèles

# Annexes

On peut configurer les arguments de la JVM pour réduire l'allocation mémoire  
**CEPENDANT**

Il faut connaître les limites de l'application (**OutOfMemoryError**)

## Heap size options

JVM PARAMETER	DESCRIPTION
-Xms	Sets the initial heap size
-Xmx	Sets the maximum heap size
-XX:MinHeapFreeRatio	Sets the minimum percentage of free space after garbage collection
-XX:MaxHeapFreeRatio	Sets the maximum percentage of free space after garbage collection
-XX:MaxDirectMemorySize	Sets the limit for the memory allocated to direct byte buffers

## RAM consumption

JVM PARAMETER	DESCRIPTION
-XX:MaxRAM	Sets the max. amount of total memory used by the JVM
-XX:MaxRAMFraction	Sets the RAM limit for JVM in fractions
-XX:MaxRAMPercentage	Sets the RAM limit for JVM per cent