Pour voir les différences entre les MPM de manière **très visible**, il est essentiel de concevoir des scénarios de test qui mettent en avant les points faibles et forts de chaque MPM. Voici des étapes concrètes et des scénarios spécifiques pour rendre ces différences évidentes.

---

1. Augmenter la charge et la concurrence

Pourquoi ?

Les MPM réagissent différemment lorsque le serveur subit une charge élevée en termes de requêtes simultanées et de concurrence. `prefork` est limité par sa consommation mémoire, tandis que `worker` et `event` utilisent des threads pour mieux gérer des charges importantes.

Comment ?

- Effectuez un test avec un grand nombre de requêtes (par exemple, **10000 requêtes simultanées avec 1000 de concurrence**).
- Commande :

```
ab -n 10000 -c 1000 http://192.168.1.186:80/test.html
```

- Résultat attendu :
  - **MPM prefork :** Augmentation du temps moyen par requête et possibles échecs (`Failed requests`) en raison de la limite mémoire par processus.
  - **MPM worker et event :** Meilleure performance grâce aux threads, avec moins d'échecs et un temps moyen plus stable.

---

2. Tester avec des connexions persistantes (Keep-Alive)

Pourquoi ?

Le MPM `event` est spécifiquement conçu pour gérer efficacement les connexions persistantes, contrairement à `prefork` et `worker`.

Comment ?

- Activez Keep-Alive dans Apache (`/etc/apache2/apache2.conf`) :

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
```

- Effectuez un benchmark en activant Keep-Alive avec `ab` :

```
ab -k -n 5000 -c 500 http://192.168.1.186:80/test.html
```

- Résultat attendu :
  - **MPM prefork :** Moins performant car chaque connexion persistante nécessite un processus dédié.
  - **MPM worker :** Performances acceptables, mais moins optimales que `event`.
  - **MPM event :** Excellente performance, car il utilise des threads non-bloquants pour gérer les connexions longues.

---

3. Tester des fichiers volumineux

Pourquoi ?

Le traitement de fichiers volumineux augmente la charge sur le CPU et la mémoire, révélant les différences dans la gestion des ressources.

Comment ?

- Ajoutez un fichier volumineux (par exemple, 10 Mo) dans `/var/www/html/` :

```
dd if=/dev/zero of=/var/www/html/large.html bs=1M count=10
```

- Effectuez un benchmark :

```
ab -n 1000 -c 100 http://192.168.1.186:80/large.html
```

- Résultat attendu :
  - **MPM prefork :** Temps moyen par requête élevé et augmentation de l'utilisation de la mémoire.
  - **MPM worker et event :** Gestion plus efficace des ressources, avec un meilleur débit (`Transfer rate`).

---

4. Tester sous contrainte mémoire

Pourquoi ?

`prefork` consomme beaucoup de mémoire, tandis que `worker` et `event` utilisent des threads, ce qui réduit la consommation globale.

Comment ?

- Limitez la mémoire disponible pour Apache avec `ulimit` :

```
ulimit -v 500000 # Limite de 500 Mo pour les processus
```

- Redémarrez Apache et lancez un test standard :

```
ab -n 1000 -c 100 http://192.168.1.186:80/test.html
```

- Résultat attendu :
  - **MPM prefork :** Échecs fréquents (`Failed requests`) en raison du manque de mémoire.
  - **MPM worker et event :** Meilleure résilience grâce à l'utilisation de threads.

---

5. Mesurer l'utilisation des ressources système

Pourquoi ?

Chaque MPM utilise différemment les ressources CPU et mémoire, en fonction de leur architecture.

Comment ?

- Pendant un benchmark, surveillez les ressources système avec :

```
htop
```

Ou :

```
vmstat 1
```

- Points à observer :
  - **Utilisation CPU (%)** : Prévisible plus élevée pour `prefork` en raison des processus multiples.
  - **Utilisation mémoire (Mo)** : Plus élevée pour `prefork`, modérée pour `worker`, et optimisée pour `event`.

Résultat attendu :

- **MPM prefork :** Haute consommation mémoire pour les processus.
- **MPM worker :** Utilisation équilibrée.
- **MPM event :** Meilleure efficacité pour les connexions longues.

---

6. Simuler des erreurs (Stress Test)

Pourquoi ?

Cela permet de voir comment chaque MPM se comporte lorsqu'il atteint ses limites (par exemple, `MaxRequestWorkers`).

Comment ?

- Réduisez le paramètre `MaxRequestWorkers` dans la configuration du MPM :
  - Pour `prefork` :

```
MaxRequestWorkers 50
```

  - Pour `worker` et `event` :

```
MaxRequestWorkers 100
```

- Lancez un benchmark dépassant cette limite :

```
ab -n 5000 -c 500 http://192.168.1.186:80/test.html
```

- Résultat attendu :
  - **MPM prefork :** Beaucoup d'échecs (`Failed requests`) après avoir atteint `MaxRequestWorkers`.
  - **MPM worker et event :** Meilleure gestion de la charge.

---

7. Monitoring avancé avec outils externes

Pourquoi ?

Les outils comme **Prometheus** et **Grafana** peuvent visualiser les différences en termes de latence, débit, et utilisation des ressources.

Comment ?

- Installez un exportateur Apache (par exemple, **apache_exporter**) :

```
sudo apt install prometheus-apache-exporter
```

- Configurez Grafana pour visualiser les métriques de chaque MPM sous charge.

Résumé des scénarios pour des différences visibles :

Test	Différence Observée	Pourquoi ?
Charge élevée	`prefork` échoue plus vite, `event` résiste mieux	Différence de gestion mémoire et threads.
Connexions persistantes	`event` est largement supérieur	Conçu pour les Keep-Alive.
Fichiers volumineux	`prefork` consomme plus de mémoire	Processus vs threads.
Contraintes mémoire	`prefork` échoue plus tôt	Haute consommation mémoire.
Surveiller les ressources	`prefork` utilise plus de CPU et mémoire	Processus multiples vs threads.
Stress test	`event` gère mieux la surcharge	Architecture optimisée.