Probabilités Appliquées : simulation de serveur

William Schmitt

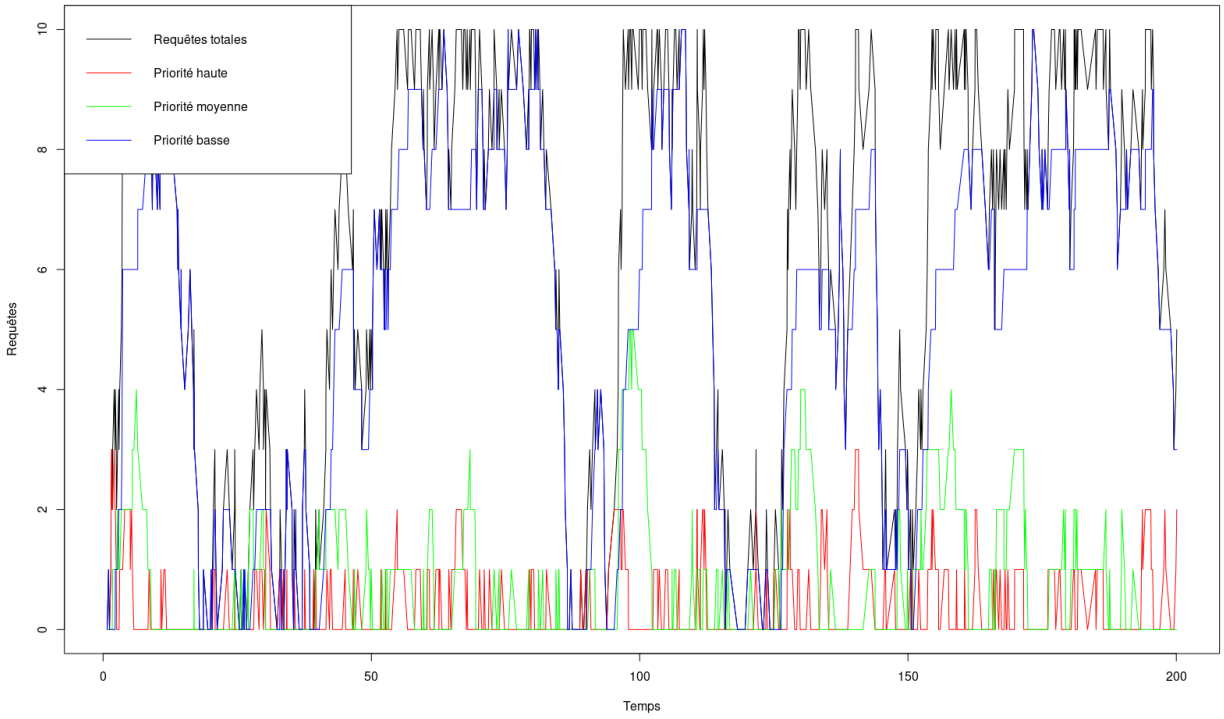
6 juin 2018

Introduction

Le projet a été réalisé avec *R*. Plusieurs choix importants ont été effectué :

- Quel que soit l'état de la file, on traite toujours une requête prioritaire avant les autres, même si elle vient d'arriver.
- Une requête arrivant vers le(s) serveur(s) est rejetée dès lors que la limite de capacité est atteinte. Cela implique qu'on ne cherche pas à supprimer une requête de priorité faible pour faire de la place à une requête prioritaire arrivant sur le(s) serveur(s).
- S'il y a plusieurs serveurs, ceux-ci sont placés en parallèle.
- La valeur du paramètre λ impacte les n serveurs. Les temps de service suivent alors des lois de même paramètre.

Résultats d'une petite simulation (1 serveur)



La simulation a tourné pour les valeurs suivantes :

- $\lambda = 2$
- $\mu = 1,9$
- `duration` $= 200$, de façon à permettre plus de lisibilité
- 1 serveur

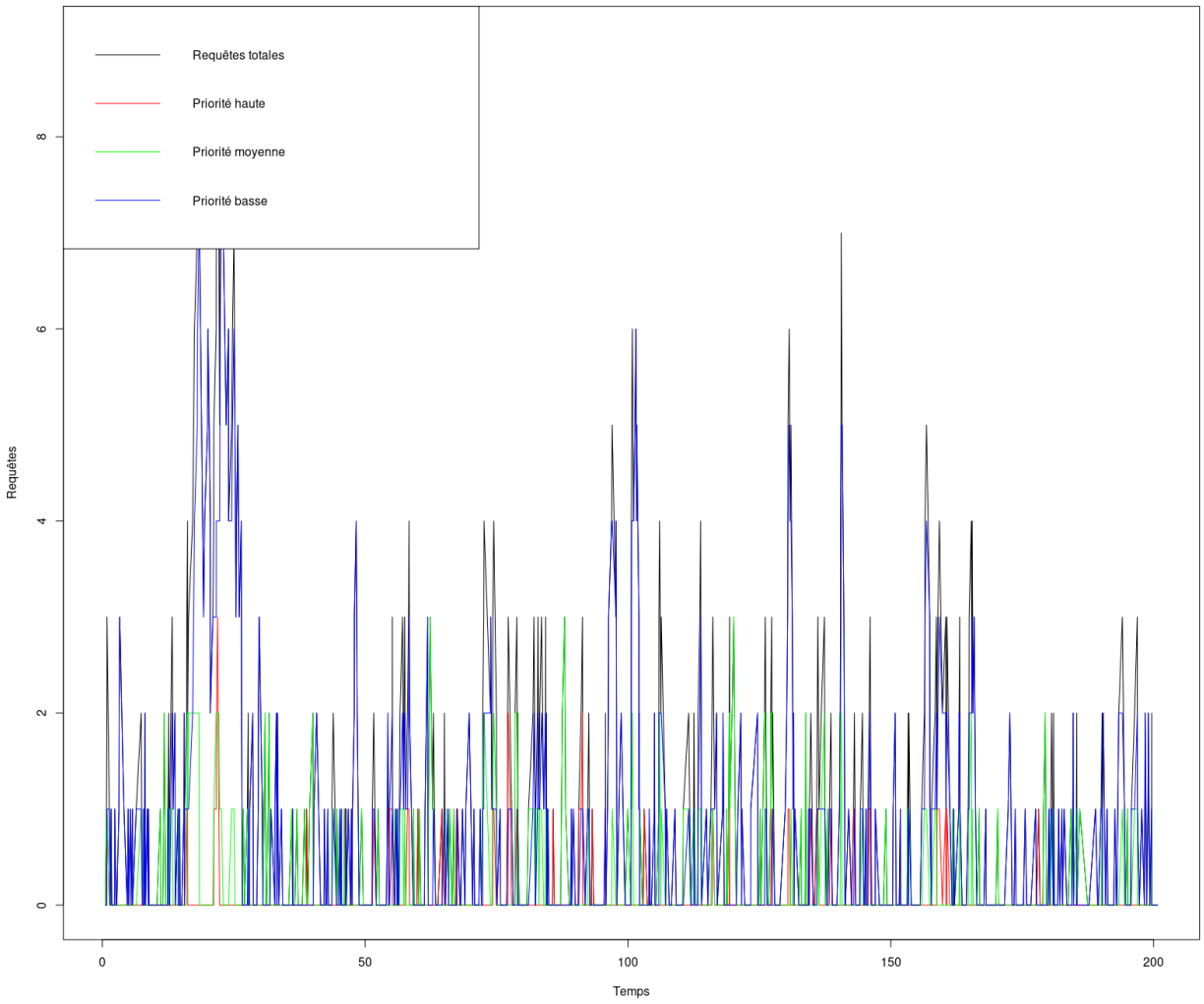
- Capacité de chaque serveur : 10
- Probabilité pour chaque priorité (des plus prioritaires aux moins prioritaires) : `0.3 0.3 0.4`

Cette représentation graphique permet de constater que les requêtes prioritaires sont traitées avant les requêtes moins prioritaires, le service fonctionne correctement, cela se vérifie par les statistiques données en fin de traitement (si `stats = TRUE`) :

```
"Moyenne des priorités hautes 0.391194968553459"
"Moyenne des priorités moyennes 0.725786163522013"
"Moyenne des priorités basses 4.99496855345912"
```

Simulation à deux serveurs

Dans cette seconde simulation, les paramètres d'entrée sont les mêmes, mis à part pour `serverCount`, auquel on affecte 2.

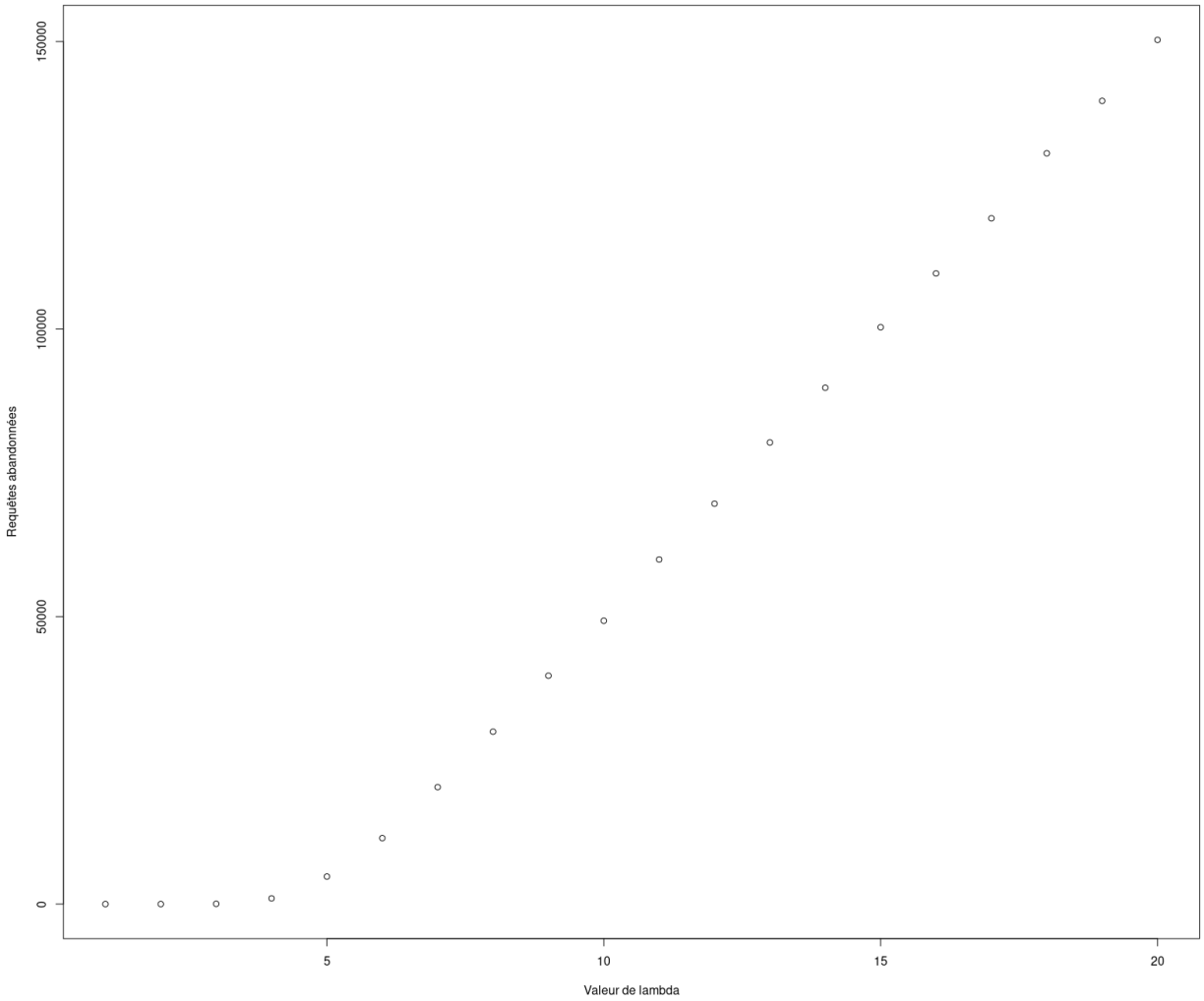


Le comportement observé est complètement différent : la file d'attente ne se remplit pas et le traitement en parallèle est très efficace. Le taux d'abandon atteint zéro.

Meta-simulations : impact des paramètres

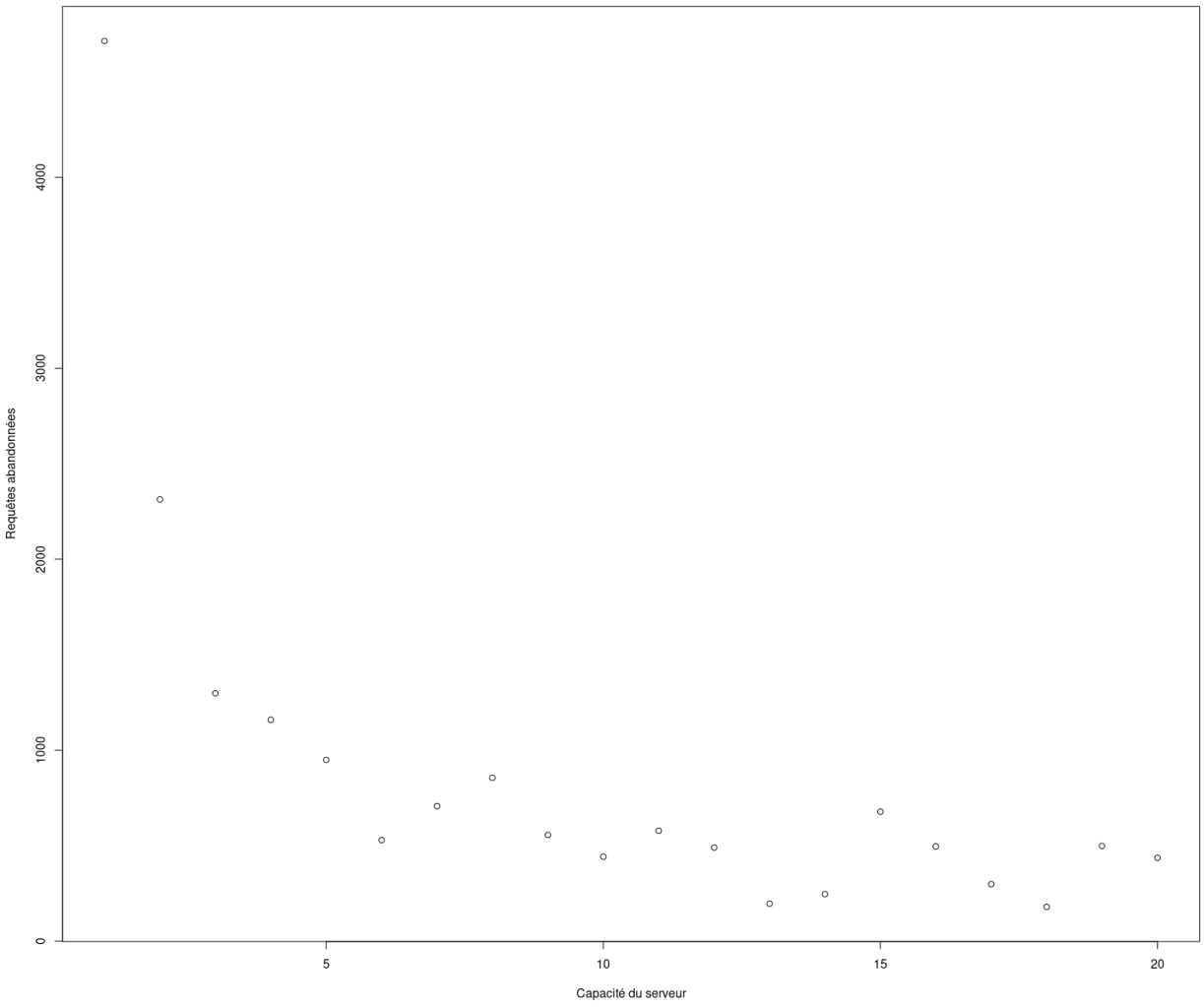
Impact de lambda

On fixe arbitrairement $\mu = 1$, et on fait varier λ de 1 à 20. La capacité est fixée à 10 et il n'y a qu'un serveur pour traiter les différentes requêtes.



Dès que le rapport $\frac{\lambda}{\mu}$ dépasse 5, on constate que la quantité de requêtes abandonnées explose.

Impact de la capacité d'un serveur



La capacité du serveur a également un impact fort sur le nombre de requêtes abandonnées. Pour faciliter les simulations, leur durée est faible. Cela explique les anomalies statistiques. La tendance semble être exponentiellement décroissante.

Choix techniques

Économie de mémoire et CPU pour les statistiques

Les tableaux permettant de générer les graphiques et les moyennes sont alloués statiquement en début de traitement grâce aux paramètres `lambda`, `mu` et `duration`. La concaténation de tableaux dans *R* est en effet peu performante, puisque concaténer une valeur à un tableau implique que le moteur **recopie l'intégralité du tableau** avant d'ajouter la nouvelle valeur. Pour des traitements longs, comme possible dans cette simulation, les chutes de performances sont énormes.

Nombre de serveurs et capacité

Les valeurs `serverCount` et `serverCapacity` sont combinés en un seul indicateur, dans la fonction `effectiveCapacity`. Cela revient à dire que les deux situations suivantes sont équivalentes :

Valeurs	Deux serveurs, capacité k	Un serveur, capacité 2k
Capacité	k	2k
Nombre de serveurs	2	1
Capacité effective	2*k = 2k	1*2k = 2k

Les deux valeurs sont modifiables indépendamment.