Simulation d'un serveur

Louis KLEIN

June 5, 2018

Ce projet a pour but de simuler un(des) serveur(s) recevant des requetes. Les requetes arrivent toutes les $1/\lambda$ unités de temps, et sont traités en moyenne au bout de $1/\mu$ unités de temps par le serveur. Les requetes non immediatement traitées, sont stockées dans une file d'attente (ou queue), de capacité N. Les requetes sont classés par priorité $1, ...n$, avec 1 décrivant les requetes les plus prioritaires. Une requete de catégorie i a une probabilité p_i d'arriver au serveur.

Hypothèses

Pour la simulation, on a fait les hypothèses suivantes : - Les durées d'arrivées des requêtes sont caractérisées par une variable aléatoire X suivant une loi exponentielle de paramètre λ - Les temps de traitements des requêtes sont caractérisées par une variable aléatoire Y suivant une loi exponentielle de paramètre μ - Une requête arrivant lorsqu'un serveur est disponible ne passe pas par la queue : elle est immédiatement traitée (si elle est de prioritée haute, ou si la queue est vide) - Si la queue est pleine, qu'importe la priorité de la requete : elle est rejetée. - les serveurs traitent les requêtes en parrallèles.

Specificités d'implantations

Generer les données

On sait que n arrivées d'une loi exponentielle sur un intervalle $[0, t]$ est distribué uniformément, selon une variable caracterisée par $Unif(0, t)$. On a donc choisi de simuler les temps d'arrivée et de traitement à l'aide de loi Uniforme.

- Pour les temps d'arrivées : comme en moyenne, on doit avoir $n = \lambda t$ arrivées (cf l'esperance de la loi $\Gamma(t, \lambda)$), on utilise la fonction `runif(lambda*t, 0, t)` de R pour le simuler.
- Pour les temps de traitement : comme en moyenne, on doit avoir $n = \mu t$ traitement, similairement, on utilise la fonction `runif(mu*t, 0, t)`.

Gestion multi-serveurs

La génération des données étant statique, on doit génerer, pour chaque serveur tous les temps de traitement hypothétiques. Même si ceux ci ne seront pas forcément utilisés (si le serveur ne traite pas de requête à ce moment là.). Pour savoir si un temps de fin de traitement doit être pris en compte, on utilise une variable qui nous indique quels serveurs sont actifs, `server_level`. Si le niveau du serveur est supérieur a cette varible, alors le temps de traitement est ignoré.

Traitements des données

On regroupe tous les temps d'arrivées et de traitement dans un `data.frame`, que l'on trie selon le temps dans l'ordre croissant. C'est sur ce `data.frame` qu'on va itérer pour traiter les données. Ainsi :

- Si c'est un temps d'arrivée de requête : on verifie si un serveur est disponible et s'il n'y a pas de requete plus prioritaire dans la queue. Si les deux conditions sont remplies, on traite la requête. Sinon, elle s'ajoute a la queue.

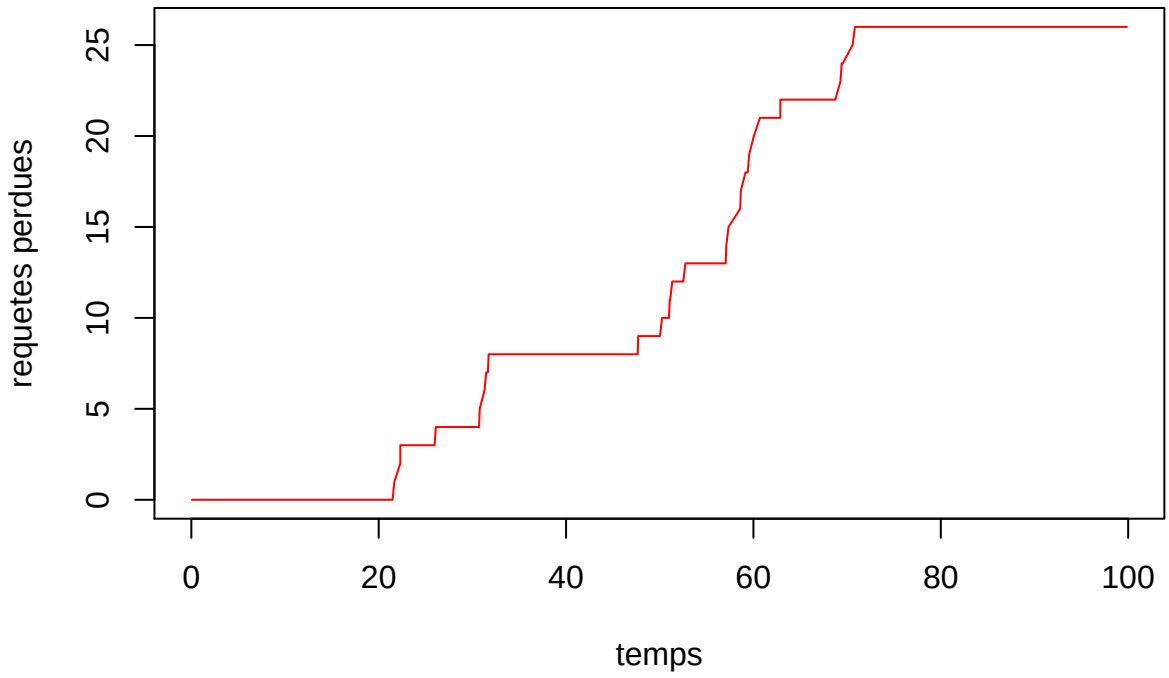
- Si c'est un temps de fin de traitement : si la queue est vide, on notifie que le serveur est disponible, sinon on traite une requête prioritaire.

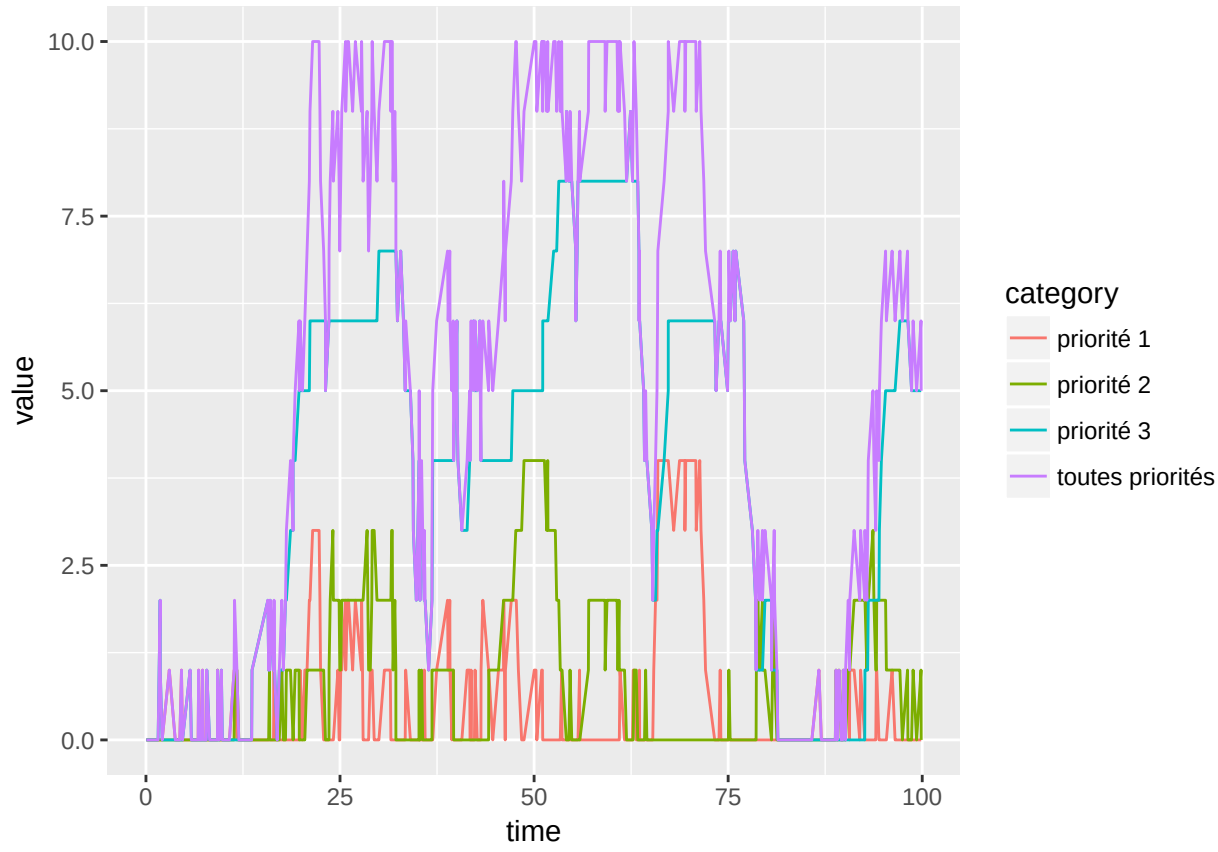
Résultats

Sur une petite simulation, avec les paramètres suivants :

- $\lambda = 2$,
- $\mu = 1.9$,
- $t = 100$ unités de temps,
- $N_{max} = 10$ requetes,
- $n_{server} = 1$,
- $p_1 = p_2 = p_3 = \frac{1}{3}$

On peut observer les résultats suivants :






```
## [1] "STATISTIQUES"
## [1] "#############"
## [1] "[ SIMULATION ] Moyenne d'utilisation de la queue : 4.86153846153846"
## [1] "[ SIMULATION ] Moyenne du nombre de requete de priorité 1 dans la queue : 0.448717948717949"
## [1] "[ SIMULATION ] Moyenne du nombre de requete de priorité 2 dans la queue : 0.738461538461539"
## [1] "[ SIMULATION ] Moyenne du nombre de requete de priorité 3 dans la queue : 3.67435897435897"
## [1] "Moyenne du nombre de requêtes dans le système :"
## [1] "[ SIMULATION ] 9.48"
## [1] "[ THEORIQUE  ] 7.71336400113897"
## [1] "Moyenne du nombre de requêtes perdues : "
## [1] "[ SIMULATION ] 0.13"
## [1] "[ THEORIQUE  ] 0.115955497912715"
```

On voit donc que meme avec un petit écart entre λ et μ, le nombre de requête perdue est déjà significatif.

Garantir un bon traitement

On s'aperçoit que pour aucune requête ne soit perdue, il faut que $\mu > \lambda$. Dans le cas ou plusieurs serveurs sont utilisés, de paramètre $\mu_1, ..., \mu_n$, alors, il faut que $\sum_{i=1}^{n} \mu_i > \lambda$.