# Projet : File M/M/1

Rapport

Dans le cadre du cours d'évaluation des performances, en première année de master SIRIS, il nous proposé un projet ayant pour objectif de créer un simulateur pour une file de type M/M/1. Ce simulateur nous permettra d'estimer le comportement d'une telle fille en faisant varier les paramètres. Je vais d'abord vous présenter l'environnement de ma simulation, puis nous allons étudier deux choses : premièrement l'écart aux valeurs théoriques et l'impact des deux variables lambda et mu. (Lambda réglant la durée entre deux arrivée, et mu réglant la durée de service). En second lieu, je vais me pencher sur les performances de mon simulateur.

## Environnement de la simulation

Le simulateur était codé en Java. Toutes les mesures ont été réalisées sur le serveur *Turing* de l'université de Strasbourg, sur Java 11

Les données ont été récupérés à l'aide d'un script bash. Elles sont renseignées dans le fichier *data.csv*. Les graphiques ont étés générés avec R grâce au script *plot.R*

## Analyse des résultats

J'ai fait une série de mesures en faisant varier les 2 paramètres :

- Lambda
- Mu

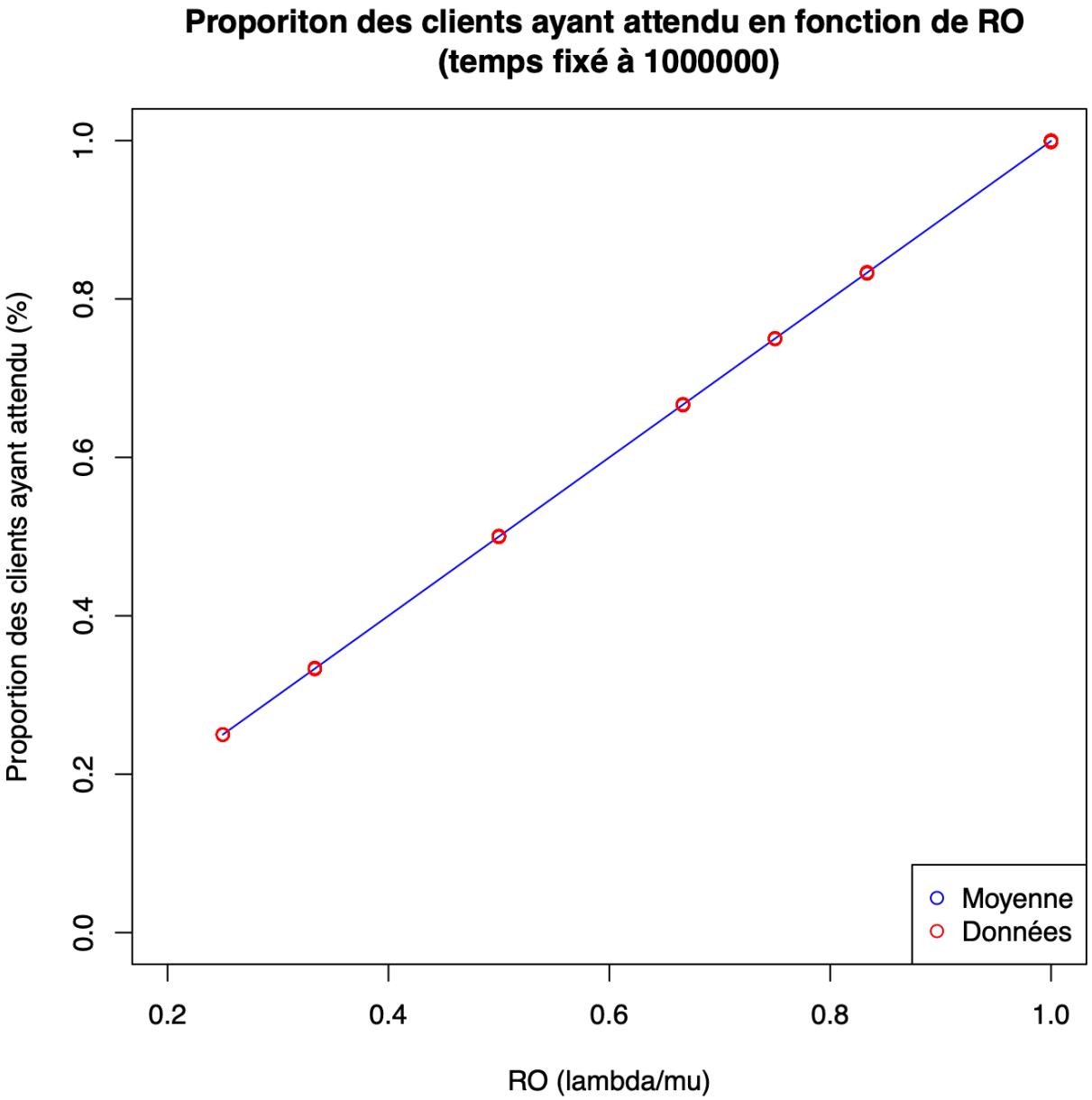
Les valeurs ont pris successivement les valeurs 2,4,6 et 8, avec lambda<mu. Le temps de simulation à été fixé à 1 00 000.

J'ai aussi fait une autre série de mesures avec les valeurs de lambda et de mu fixés à *lambda=5* et *mu=6,* mais en faisant varier le temps de simulation afin d'étudier l'écart à la théorie en fonction du temps de simulation. Les valeurs de temps seront 1, 10, 100 1 000, 10 000, 100 000, 1 000 000 et 10 000 000.

Afin de diminuer les écarts statistiques, j'ai répété 10 fois chaque mesure. Évidemment, il aurait été préférable de choisir un nombre de répétition plus élevés, mais comme un lancement de la simulation peut durer jusqu'à 15 secondes, il est préférable de diminuer les répétitions afin de pouvoir obtenir les résultats dans un temps raisonnable.

## Impact de Ro sur les clients avec attente

A mon sens, le caractère le plus important est de permettre de servir le client sans attente (la proportion des clients arrivant avec une file vide). Le graphe suivant montre l'évolution de cette proportion en fonction de Ro (lambda/mu)




Comme prévu, la proportion des clients ayant attendu augmente à mesure que lambda se rapproche de mu. En effet, le temps de service étant égal au temps entre l'arrivée de deux clients, ceux-ci arrivent souvent avec une file pleine.

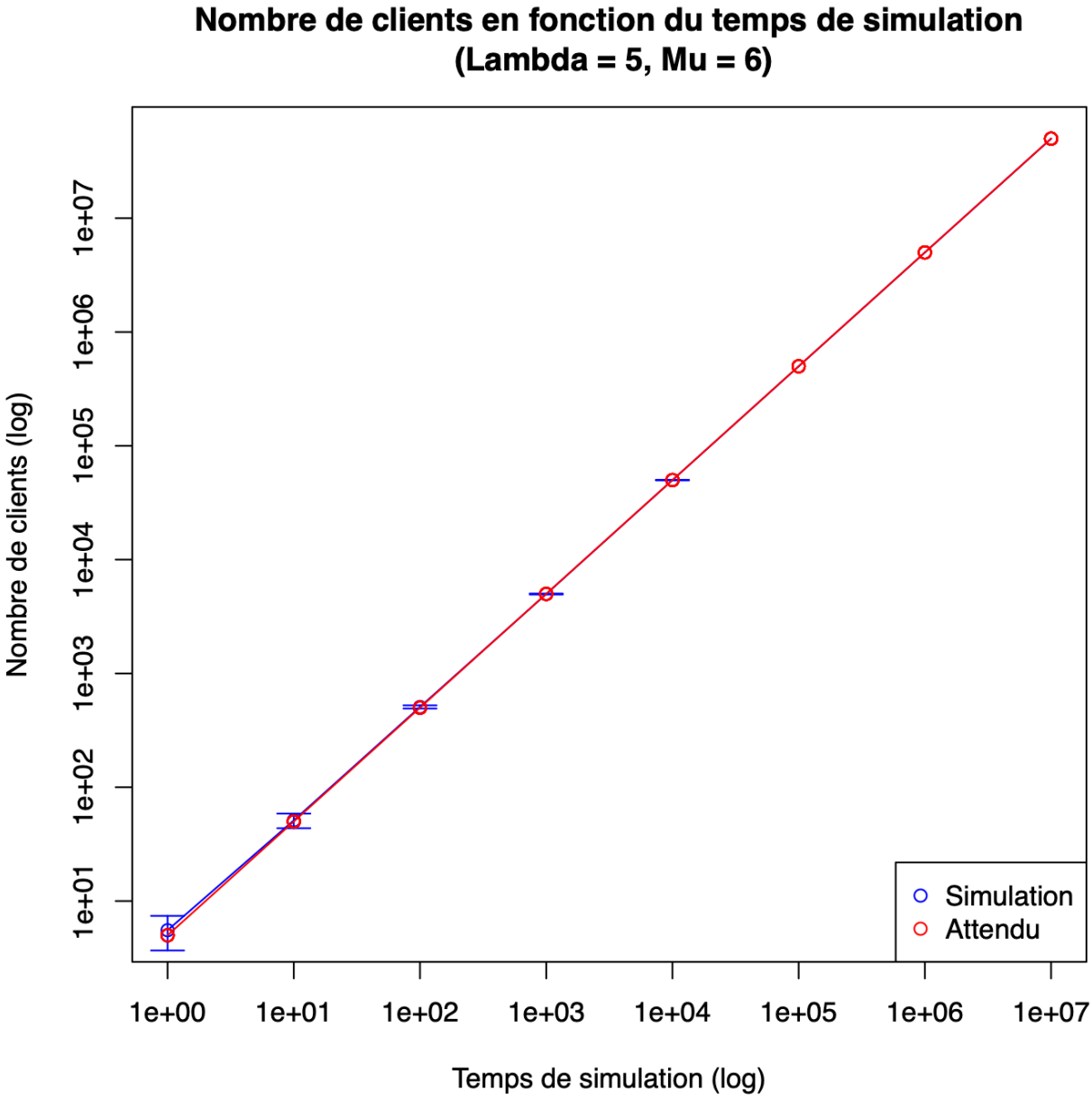
Je n'ai pas fait de simulation avec lambda > mu, car cela augmentait considérablement le temps d'exécution de la simulation. Mais on peut présager que dans ce cas la taux de clients ayant attendu serait extrêmement proche de 1.

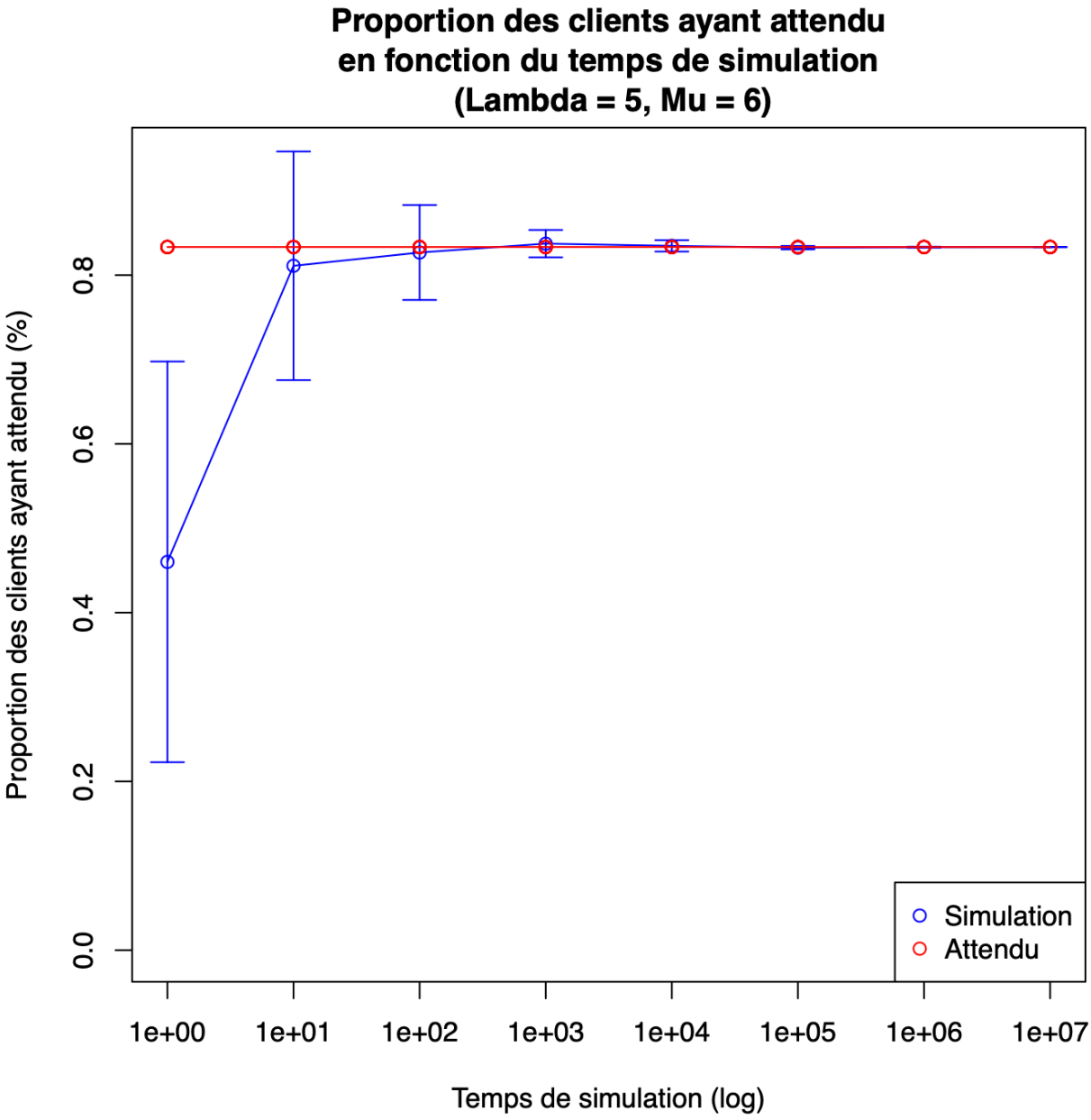
## Écart avec la théorie

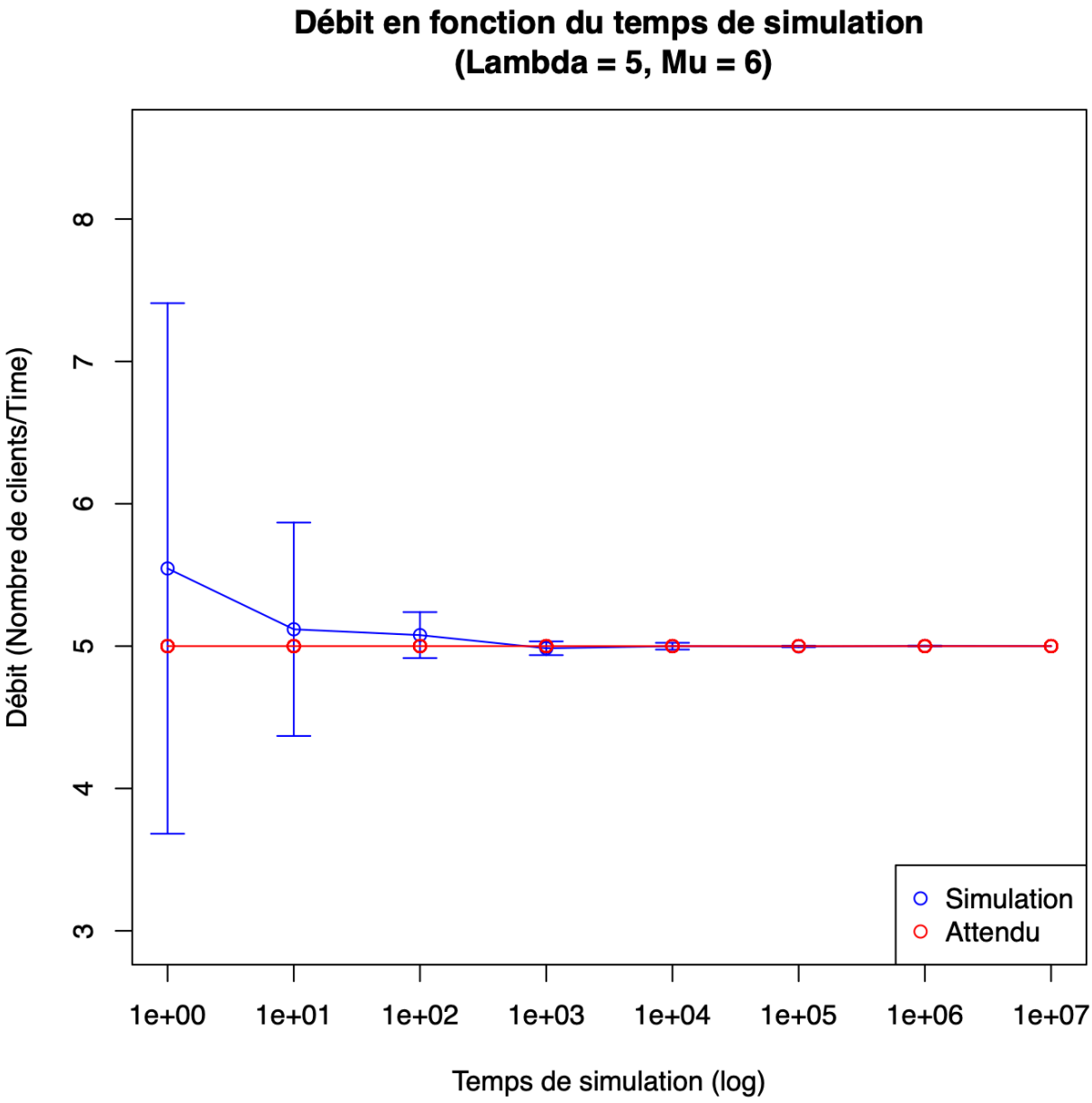
Le coeur de ce projet est de créer un simulateur. Le simulateur est bon si il s'approche des valeurs attendue. Nous allons donc comparer les valeurs théoriques avec les valeurs obtenues pour les variables suivantes :

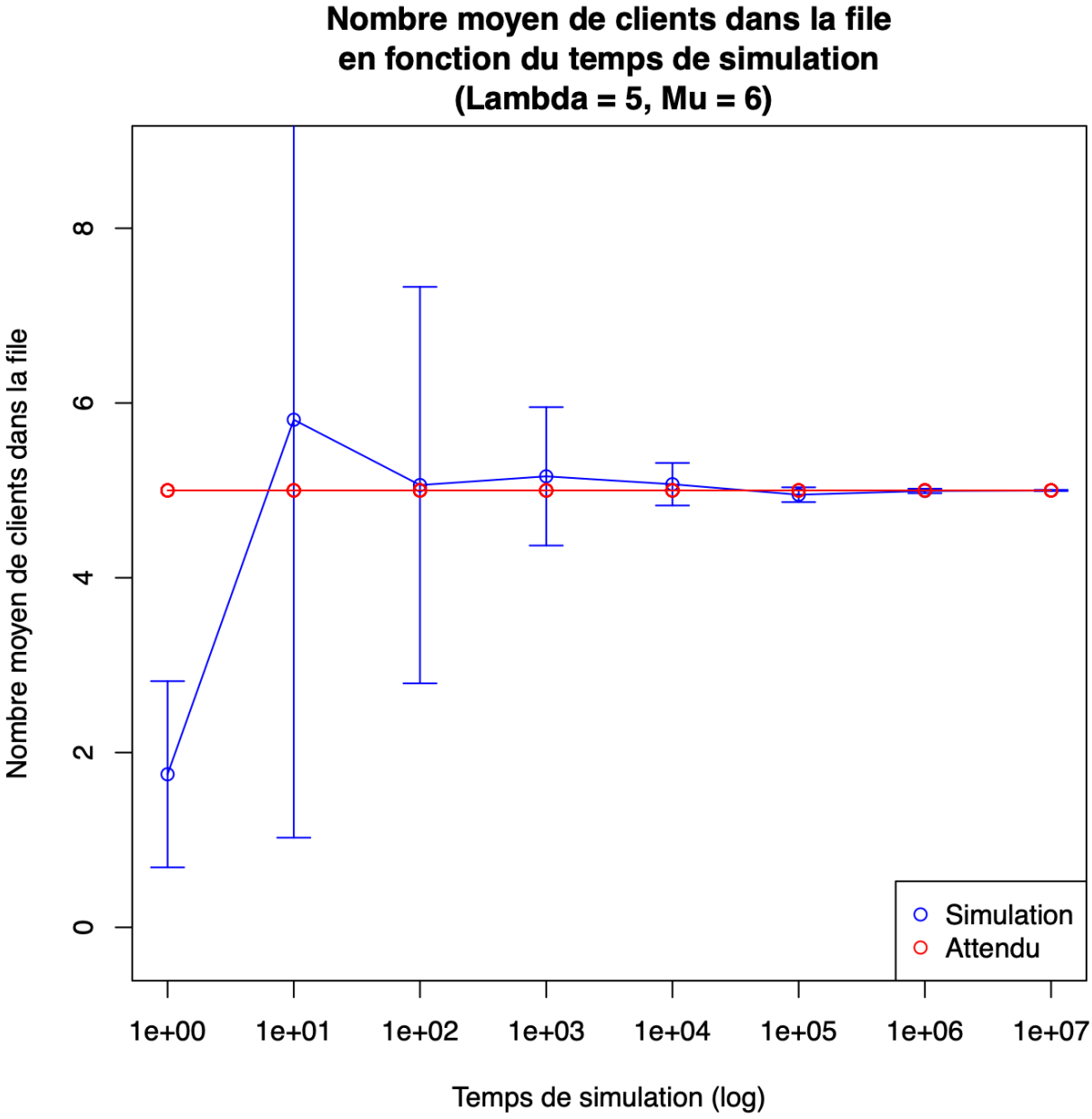
- Nombre total de client
- Proportion des clients sans attente
- Débit
- Nombre moyen de client dans la file
- Temps moyen de séjour dans la file

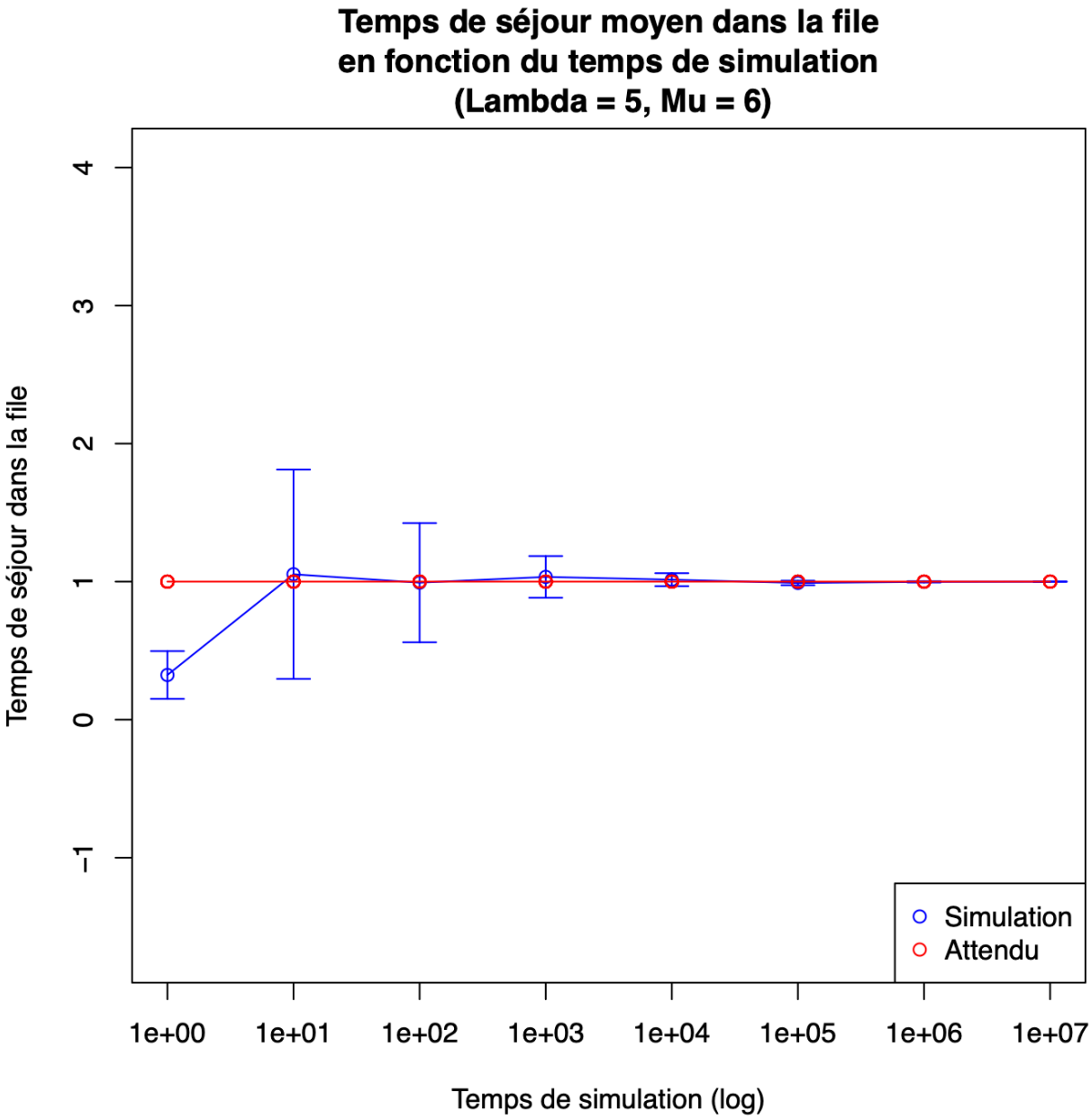
Pour ceci, le temps de simulation variera de 1 à 10 000 000, et les valeurs lambda et mu resterons fixes (lambda = 5 et mu = 6).




Proportion des clients ayant attendu
en fonction du temps de simulation
(Lambda = 5, Mu = 6)
Proportion des clients ayant attendu (%)
0.0
0.2
0.4
0.6
0.8
1e+00
1e+01
1e+02
1e+03
1e+04
1e+05
1e+06
1e+07
Temps de simulation (log)
Simulation
Attendu

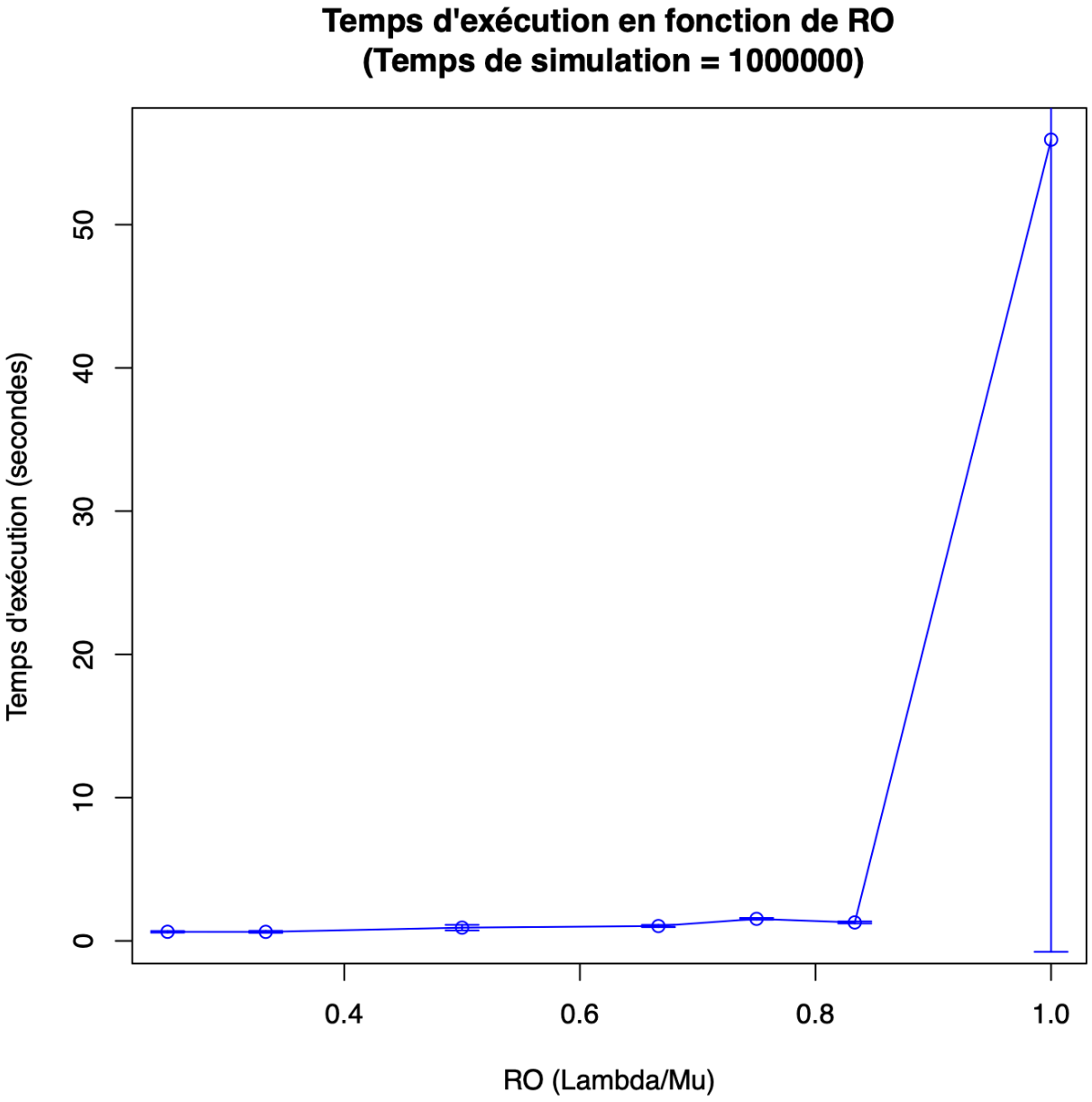

Débit en fonction du temps de simulation
(Lambda = 5, Mu = 6)
Débit (Nombre de clients/Time)
3
4
5
6
7
8
1e+00
1e+01
1e+02
1e+03
1e+04
1e+05
1e+06
1e+07
Temps de simulation (log)
Simulation
Attendu


Nombre moyen de clients dans la file
en fonction du temps de simulation
(Lambda = 5, Mu = 6)
Nombre moyen de clients dans la file
0
2
4
6
8
1e+00
1e+01
1e+02
1e+03
1e+04
1e+05
1e+06
1e+07
Temps de simulation (log)
Simulation
Attendu




Comme vu dans les 5 graphiques précédents, la simulation semble très bien s'approcher des valeurs théoriques quand le temps de simulation devient grand (supérieur à 1000). Les écarts types, très élevés au début deviennent extrêmement petits lorsqu'on augmente le temps de simulation. On peut dire que le simulateur est fidèle à la théorie quand le temps de simulation devient significativement élevé.

## Performances du simulateur




Comme prévu, on peut voir que le temps d'exécution du simulateur augmente lorsque Ro augmente. En effet, un lambda égal à mu créera beaucoup plus d'insertion dans la file et cela est très chronophage (car il faut parcourir la file pour trouver ou insérer le prochain événement). En revanche, si la liste est vide, le simulateur ne parcourra pas la file, mais insérera l'événement directement au début. Ainsi, si la file est souvent vide, le simulateur s'exécutera beaucoup plus vite.

L'immense écart type que l'on voit pour Ro = 1 s'explique facilement par le fait que le temps de calcul dépend des valeurs de lambda et mu. En effet, lambda et mu = 2

donnera un temps de calcul beaucoup moins long (moins d'arrivée en file non vide) que si lambda et mu = 8.

On peut aussi supposer que si Ro devient supérieur à 1, les temps d'exécutions augmenteraient drastiquement, car la file serait toujours pleine, et très rapidement extrêmement longue, et il faudrait beaucoup plus de temps pour la parcourir.

Afin d'améliorer les performances de mon simulateur, une bonne piste serait de ne pas utiliser de boucle *foreach* pour la recherche dans la liste. Java possède des outils plus performant, comme par exemple *Iterator.*

Cependant, mon simulateur reste assez bon, car il s'exécute en 10 secondes sur *turing* avec les paramètres lambda=5, mu=6, durée=10 000 000)