8INF802 – Simulation de systèmes
Rapport de projet

Sacha GUYOT
GUYS07119908

Alan Guivarch
GUIA24119800

Scénario à un seul ascenseur

Dans toutes les expériences qui suivent, le processus d'arrivée de Poisson a un taux d'arrivée moyen de lambda = 0.5 personnes par minute, et le temps de travail suit une distribution exponentielle avec une moyenne de 60 minutes. Les hypothèses faites sont celles données dans le sujet.

Algorithme d'ordonnancement

FCFS

Pour tester le cas de l'algorithme FCFS avec un ascenseur nous avons utilisé les arguments suivants :
elevators 1 floors 7 scheduler fcfs idle mid maxPeople 100000

On obtient un temps d'attente moyen de 223.30 secondes, soit 3.72 minutes, ou 3:43 minutes.

SSTF

Pour tester le cas de l'algorithme SSTF, nous avons utilisé les arguments suivants :
elevators 1 floors 7 scheduler sstf idle mid maxPeople 100000

On obtient un temps d'attente moyen de 194.53 secondes, soit 3.24 minutes, ou 3:15 minutes.

Linear Scan

Pour tester l'algorithme Linear Scan, nous avons utilisé les arguments suivants :
elevators 1 floors 7 scheduler ls idle mid maxPeople 100000

On obtient un temps d'attente moyen de 205.74 secondes, soit 3.43 minutes, ou 3:26 minutes.

On remarque que l'algorithme du "Shortest-Seek-Time-First" permet aux utilisateurs d'avoir un temps d'attente plus court que les deux autres algorithmes.

Politique de ralenti

Pour les expériences suivantes, nous avons utilisé l'algorithme le plus efficace des expériences précédentes, c'est-à-dire l'algorithme SSTF.

Arrêt à l'étage supérieur

Pour tester le cas de la politique de ralenti où l'ascenseur retourne à l'étage supérieur lorsqu'il n'y a plus aucun utilisateur, nous avons utilisé les arguments suivants :
elevators 1 floors 7 scheduler fcfs idle high maxPeople 100000

On obtient un temps d'attente moyen de 203.59 secondes, soit 3.39 minutes, ou 3:24 minutes.

Arrêt à l'étage central

Pour tester le cas de la politique de ralenti où l'ascenseur retourne à l'étage central lorsqu'il n'y a plus aucun utilisateur, nous avons utilisé les arguments suivants : elevators 1 floors 7 scheduler fcfs idle mid maxPeople 100000

On obtient un temps d'attente moyen de 194.14 secondes, soit 3.24 minutes, ou 3:14 minutes.

Arrêt à l'étage inférieur

Pour tester le cas de la politique de ralenti où l'ascenseur retourne au rez-de-chaussée lorsqu'il n'y a plus aucun utilisateur, nous avons utilisé les arguments suivants :
elevators 1 floors 7 scheduler fcfs idle low maxPeople 100000

On obtient un temps d'attente moyen de 195.15 secondes, soit 3.25 minutes, ou 3:15 minutes.

On remarque que la politique du retour de l'ascenseur à l'étage central permet aux utilisateurs d'avoir un temps d'attente plus court que la politique du retour à l'étage supérieur. La différence entre le retour à l'étage central et le retour au rez-de-chaussée est très petite.
Nous avons donc essayé d'augmenter le nombre d'étages pour voir si cela permettait de trancher plus sûrement entre ces deux options. En passant de 7 à 20 étages, on obtient un temps d'attente moyen de 3830.86 secondes pour le retour au rez-de-chaussée et 3781.55 secondes pour un retour à l'étage central. On obtient donc une différence plus marquée en faveur du retour à l'étage central.

Scénario à plusieurs ascenseurs

Pour les expériences suivantes, nous avons utilisé l'algorithme d'ordonnancement et la politique de ralenti les plus efficaces des expériences précédentes, c'est-à-dire l'algorithme SSTF et la politique de retour à l'étage central.

Deux ascenseurs

Pour tester ce cas, nous avons utilisé les arguments suivants :
elevators 2 floors 7 scheduler fcfs idle mid maxPeople 100000

On obtient un temps d'attente moyen de 124.85 secondes, soit 2.08 minutes, ou 2:05 minutes.

Trois ascenseurs

Pour tester ce cas, nous avons utilisé les arguments suivants :
elevators 3 floors 7 scheduler fcfs idle mid maxPeople 100000

On obtient un temps d'attente moyen de 117.02 secondes, soit 1.95 minutes, ou 1:57 minutes.

On remarque que l'impact le plus important est visible lorsque l'on passe d'un ascenseur à deux. On passe de 3:14 minutes d'attente à 2:05 minutes, et on réduit donc le temps d'attente de plus d'un tiers.
En ajoutant un troisième ascenseur, la différence de temps d'attente est beaucoup moins importante (moins de 8 secondes, soit environ 6%).

Pour répondre au problème quant à l'intérêt d'ajouter un ascenseur supplémentaire au pavillon de l'humanité, les expériences précédentes montrent que même si le temps d'attente sera réduit, une réduction de moins de 8 secondes ne semble pas justifier l'ajout d'un nouvel ascenseur.

Résumé des résultats

La politique d'ordonnancement la plus efficace est celle utilisant l'algorithme SSTF (shortest seek time first).
La politique d'attente la plus efficace est celle consistant à renvoyer l'ascenseur à l'étage central.
L'ajout d'un troisième ascenseur ne semble pas pertinent étant donné le très faible gain de temps d'attente donné par la simulation (moins de 8 secondes).

Manuel d’utilisation

Le main se trouve dans ElevatorSim.
Les arguments possibles sont les suivants :

elevators <int> ou e <int> pour indiquer le nombre d'ascenseurs.
Valeur par défaut : 1

floors <int> ou f <int> pour indiquer le nombre d’étages.
Valeur par défaut : 7

scheduler <String> ou s <String> pour indiquer le type de scheduler parmi “fcfs”, “sstf” et “ls”.
Valeur par défaut : “fcfs”

idle <String> ou i <String> pour indiquer le type de politique de ralenti entre “mid”, “high” et “low”.
Valeur par défaut : “mid”

maxPeople <long> ou mp <long> pour indiquer le nombre maximum d’utilisateurs à simuler. -1 pour illimité.
Valeur par défaut : -1

arrivalLambda <double> ou al <double> pour modifier le taux moyen d’arrivée des utilisateurs.
Valeur par défaut : 1/2

workLambda <double> ou wl <double> pour modifier le temps moyen de travail des utilisateurs.
Valeur par défaut : 1/60

speed <double> pour modifier la vitesse des ascenseurs (en étages par minute).
Valeur par défaut : 6.0

Lancer le programme sans arguments équivaut à écrire :
java -jar ElevatorSim.jar elevators 1 floors 7 scheduler fcfs idle mid maxPeople -1 arrivalLambda 0.5 workLambda 0.01667 speed 6

Le programme s’arrête tout seul lorsqu’il atteint la limite d’utilisateurs, ou lorsqu’on clic sur le bouton STOP, ou lorsqu’on tape quelque chose dans la console et qu’on appuie sur enter.

Le jar du programme se trouve dans out/artifacts/ElevatorSim_jar.
Il peut mettre quelques secondes avant de faire apparaître le bouton STOP.