Construction Objets Avancée

Giuseppe Lipari

March 11, 2019

Instructions

Vous devez rendre sur gitlab le code demandé avec un fichier README.md qui contient :

- Vos noms ;
- Pour chaque question :
  - Si vous avez réussi à coder les fonctionnalités demandées
  - La liste de tests de régression correspondants à la question

TP 5: MetaSim

On considere l'exemple sur la queue qui se trouve dans `metasim/examples/queue/`.

Question 1: temps moyenne d'attente

On voudrait mesurer le temps moyenne d'attente avant qu'une requête soit processée. C'est le temps moyenne du moment que la requête arrive dans la queue au moment que la queue termine le processing. Pour faire ça, il faut memoriser le temps d'arrivé dans la queue quelque part ...

- Concevoir une class `Message` que sera trasmis de la source à la queue et de ce dernière à la `Sink`. L'objet message memorise l'instant d'arrivé dans la queue dans une variable.
- Modifier les classes `Source`, `Queue` et `Sink` pour gérér les messages. Dans la classe Queue, vous pouvez decider d'utiliser :
  - une queue d'objets de type Message, ou
  - une queue des pointeurs à la classe Message.

  Vous pouvez choisir librement. En tout cas, **attention** à la gestion du cycle de vie de l'objet Message! Il faut éviter des fuites de memoire (*memory leaks*).

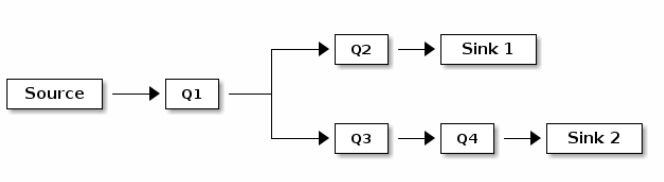
- Mesurez le temps moyenne de processing d'un message en supposant que les messages arrivent avec une distribution exponentielle de paramètre λ, et qu'ils sont servis avec temps constant inferieur à $1/\lambda$ (utiliser les classes `ExponentialVar` et `DeltaVar`).
- Testez le bon fonctionnement de la nouvelle implementation à l'aide des fonctions de debugging `sim_step()` et `run_to()`.

Question 2: réseaux de nodes

Étendre la class `Queue` pour supporter plusieurs noeuds destination. Ajouter un constructeur *template* à la classe queue qui prends en parametre 2 iterateurs à un container de pointeurs à `Node`.

Après le service, le message est envoyé à tous les noeuds destinataires.

Simuler le réseau suivant:



Mesurer les temps de parcours de la source à `Sink1` et à `Sink2`, en supposant que les messages sont produit avec distribution exponentielle avec paramàtre $\lambda = 0.5$, et les temps de service sont aussi exponentiels avec paramètres $\mu_1 = \mu_4 = 0.6$, $\mu_2 = \mu_3 = 0.75$.

Question 3: serveur multi-queue

Concevoir une class `Server` qui contient plusieurs queues. Quand un message arrive au serveur, il est envoié à une des queues à tour de rôle. Quand le message à été processé par la queue, il est envoyé au noeud suivant.

Supposons que les messages arrivent avec une distribution exponentielle de paramètre $\lambda = 0.5$, et que les queues ont un temps de service avec distribution exponentielle de paramètre $\mu = 0.6$. Mesurez le temps d'attente moyenne en fonction du nombre n de queues dans le serveur, pour $n = 1, \ldots, 8$.

Question 4: abstract factory

Utilisez l'abstract factory pour créer un réseau de noeuds à partir d'un fichier texte. Vous pouvez v'inspirer de la manière que la librairie utilise pour créer des variables alèatoires (voir randomvar.hpp et regvar.hpp).

- D'abord, créez la fonctions statique

```
static std::unique_ptr<Source> Source::createInstance(std::vector<std::string> &par);
```

pour créer un objet `Source`. Le vecteur `par` contient les parametres de l'objet.

- Faire la même chose pour les classes `Queue`, `Sink`, `Server`
- Enregistrer les fonctions dans l'abstract factory.
- Écrire un programme pour lire et faire le parsing d'un fichier txt, et créer les objets specifiés dans le fichier. Un exemple de fichier de specification:

```
Source src1(srv, 0.5)
Source src2(srv, 0.7)
Source src3(srv, 0.4)
Server srv(2, 0.8, snk)
Sink snk();
```

dans ce fichier on specifie 3 sources (avec distribution exponentielle et valeur de λ dans le deuxième paramètre) qui envoient les messages à un serveur qui contient 2 queues, avec service exponentiel et valeur de $\mu = 0.8$, et qui envoie les résultats à la sink `snk`.

Le programme lance une simulation pour calculer le temps d'attente moyenne des messages de leur source au sink et imprime le resultat sur la sortie standard.