Serveur de fichiers - 1ère partie

Miguel Cerca, Elie N'djoli

Introduction

L'objectif de ce laboratoire est de réaliser un serveur de fichier supportant le traitement de requête simultanée grâce à la concurrence et au paradigme producteur-consommateur.

L'objectif de cette première partie est d'expérimenter le code fourni. Ce dernier compile et s'exécute dès le départ. Le but est donc d'améliorer les performances de départ. Pour cela, il faut implémenter trois choses. L'interface générique *AbstractBuffer*, le *thread* de dispatching des requêtes et le *thread* de traitement de requête.

Questions/Réponses

— Comparez la performance de cette version concurrente avec celle de la version de base. Constatezvous une amélioration ? Oui, on voit une nette amélioration grâce au tampon de de requêtes. — Que se passe-t-il lorsqu'on lance un nombre de requêtes très important (par ex. 10'000), et comment l'expliquez-vous ? Une erreur bad-alloc se produit. Cela est dû au fait qu'il n'y a plus de mémoire disponible pour agrandir le QVector contenant les *threads*. Cela provoque donc une erreur d'allocation de mémoire. — Comment pourrait-on protéger le serveur de cet effet néfaste ? Pour éviter l'effet néfaste, il suffit de limiter le nombre de threads. Pour cela, il est nécessaire de mettre une taille limite au tampon de requêtes qui est le *QVector* buffer.

Description de l'implémentation

Tout d'abord, on a implémenté la classe *AbstractBuffer*. Cette dernière est générique car elle sert aussi bien aux requêtes qu'aux réponses. Elle est utilisée pour faire le tampon des requêtes et celui des réponses. Elle contient les méthodes *get()* et *put()* qui sont implémentés dans la classe *MesaBuffer*. Mesa est un type de moniteur. Sa propriété est le fait que le *thread* qui appelle le signal garde le mutex. Cela est notamment visible dans l'implémentation faite des méthodes *get()* et *put()*. On peut voir les *while*. On a donc décidé d'utiliser le moniteur Mesa comme producteur-consommateur pour cette première partie.

Une autre partie importante que l'on a dû implémenter est la classe *RequestDispatcherThread*. Cette dernière prend les requêtes dans le tampon de requêtes et les envoie au serveur grâce à la classe *SendRequestThread*. Le nom de cette dernière parle de lui-même. Elle envoie la requête au serveur. Puis, elle ajoute la réponse du serveur dans le tampon de réponses. Le fait d'avoir justement fait *AbstractBuffer* générique permet de l'utiliser aussi pour les réponses comme cela a déjà été évoqué.