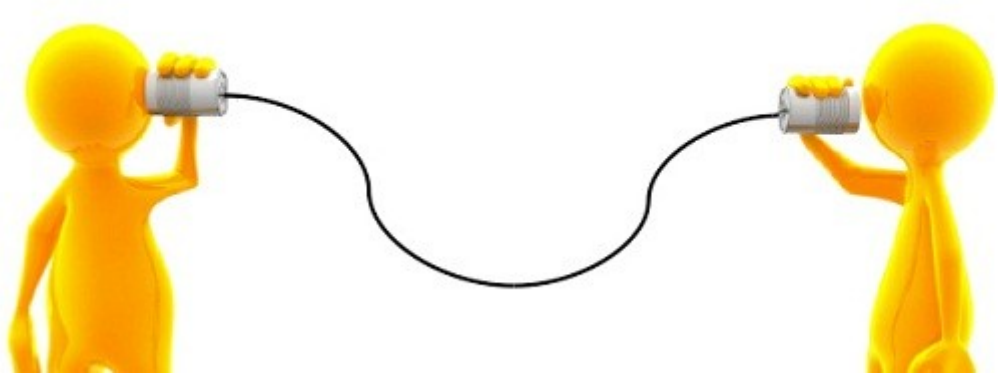
Rapport de BE

Supplément de rapport



Bonneval Fabien — **Dzieciol** Nicolas

14/12/2015

1. Corrections

Lors de la phase de conception nous avons fais l'erreur de tout tester sur un seul thread et cela à entraîner quelques soucis lors du passage au multi-thread. Le problème majeur à été une altération du pointeur sur un de nos mutex. Pour trouver cette erreur nous avons utilisés deux outils, valgrind et gdb.

Nous avons changé le flag _abo_traité qui nous permettait de savoir si l'on pouvait tenter de s'abonner par l'utilisation d'un mutex ce qui rend plus sur l'utilisation de la fonction d'abonnement.

Pour réaliser les tests nous avons mis dans le fichier main des defines que l'on doit commenter ou dé-commenter pour réaliser les différents tests unitaires ou le programme de calcul de pi.

2. Tableau de test mis a jour

№ du test	Fonction testée	Cas	État d'avancement			Résultats (OK / A)	Remarque
			T	NT	ATT		
1	**initMsg**	Appel par un thread	X			OK	
2		Appel par plusieurs thread	X			OK	
3	**aboMsg**	Appel alors que le service n'est pas initialisé	X			OK	
4		Abonnement de plusieurs thread avec id différents	X			OK	
5		Abonnement de plusieurs thread avec id identiques	X			OK	
6		Un thread s'abonne plusieurs fois avec des identifiants différents	X			OK	
7		Un thread s'abonne plusieurs fois avec des identifiants identiques	X			OK	
8		Appel de la fonction abonnement alors que le nombre maximal d'abonnés est atteins	X			OK	
9	**sendMsg**	Appel de cette fonction alors que le service n'est pas lancé	X			OK	
10		Appel de cette fonction alors que l'on est pas abonné	X			OK	
11		Envoi d'un message à un identifiant inexistant	X			OK	
12		Envoi d'un message à un identifiant existant	X			OK	
13		Envoi de message alors que l'on est pas abonné	X			OK	
14		Envoi d'un message à un destinataire qui a une boite pleine	X			OK	
15		Envoi d'un message en broadcast		X			pas codé

№ du test	Fonction testée	Cas	État d'avancement			Résultats (OK / A)	Remarque
			T	NT	ATT		
17		Réception d'un message alors que la boite est vide	X			OK	
18	recvMsgBlock	Appel de cette fonction alors que le service n'est pas lancé	X			OK	
19		Appel de cette fonction par un thread non abonné	X			OK	
20		Réception d'un message alors que la boite est vide	X			OK	
21	getNbMsg	Appel de cette fonction alors que le service n'est pas lancé	X			OK	
22		Appel de cette fonction par un thread non abonné	X			OK	
23		Appel de cette fonction par un thread abonné	X			OK	
24	desaboMsg	Appel de cette fonction alors que le service n'est pas lancé	X			OK	
25		Appel de cette fonction par un thread non abonné	X			OK	
26		Appel de cette fonction par un thread abonné	X			OK	
27		Appel de cette fonction par un thread abonné ayant encore des messages	X			A	Erreur de segmentation, free multiples
28	finMsg	Appel de cette fonction alors que le service n'est pas lancé	X			OK	
29		Appel de cette fonction sans le flag alors que le service est toujours en utilisation	X			OK	
30		Appel de cette fonction sans le flag alors que le service n'est plus utilisé	X			OK	
31		Appel de cette fonction avec le flag	X			OK	

Les tests unitaires sont plutôt satisfaisants seuls deux erreurs subsistent par rapport aux tests souhaités initialement. Nous n'avons pas eu le temps de coder la fonction permettant de faire de l'envoi de message en broadcast. L'erreur plus contraignante est l'erreur de segmentation lorsque l'on désabonne un client alors qu'il lui reste des messages à consulter, gbd et valgrind nous indique que l'on fait des libérations abusives avec un peu de temps ce problème serait résolu.

3. Résultat des tests sur scénario calcul de pi

La fonction s'exécute correctement dans la plupart des cas.

Il y a des soucis quand :

- le pas est réglé très petit par rapport au nombre d'itération (rapport nbiter/pas > 1000)
    - Le calcul s'arrête en plein milieu un des thread doit attendre un mutex ceci est probablement dû au accès très rapide aux différentes fonctions et nous n'avons pas eu le temps de trouver la source exacte du problème
- le nombre d'itération est très grand (>1.000.000.000)
    - Le programme s'exécute, mais il n'arrive pas à terminer

Quand le nombre de message maximum dans est réglé à 1 ou 2 le programme fonctionne correctement et se termine (mis a part dans les deux cas précédents).

Nous avons comparé les performances avec ce programme sur quatre thread dont trois pour le calcul avec un programme séquentiel et nous obtenons à peu près les mêmes performances (moins d'une seconde d'écart)

4. Conclusion

Le BE est dans la globalité terminée, il reste quelques cas d'erreur connus et probablement d'autres encore non décelés, nous avons eu un gros problème que nous avons mis du temps à corriger et qui nous a beaucoup retardés. Pour finir totalement le BE et assurer un bon fonctionnement nous estimons qu'il nous fraudait à peu près 10h de travail.