SR : Construction d'un mini-server web
Détail des extensions

Rémy Boutonnet
Jérémy Derdaele

3 avril 2014

Ce document présente un bref résumé du principe des solutions adoptées pour les différentes extensions réalisées.

1 Type MIME (*)

Apricot gère nativement tout type de documents (et retourne en conséquence, l'entête "content-type" adaptée).
Nous utilisons la libmagic pour déterminer le type mime du fichier à envoyer (notamment en lisant le magic number au début du fichier concerné).

2 Gestion des requêtes dynamiques : erreurs et fichier journal (***)

La gestion des requêtes dynamiques se fait de manière asynchone, c'est à dire que lorsqu'un cgi a terminé de générer les données qu'il souhaite envoyer, le worker qui a lancé le cgi va récupérer le contenu à envoyer et créer une entête http valide.
Si il y a eu un problème d'execution du programme cgi, le worker va alors renvoyer un message d'erreur au client (501 Internal Error).
Nous utilisons un fichier intermédiaire pour passer les données du cgi au worker (qui s'occupe de l'envoi) en essyant d'optimiser le plus possible les transferts (notamment en utilisant le memory mapping).
Afin d'être valide pour Apricot, un cgi doit toujours envoyer le content type des données qu'il doit envoit en premier.
L'architecture des workers nous permet par ailleurs de gérer plusieurs cgi (chaque worker possède une table de cgi qui sont en cours d'execution, cela rend le worker assez souple car il peut continuer de servir d'autre documents pendant qu'un cgi génère du contenu.

3 Interface d'administration du serveur (**)

L'interface d'administration du serveur d'Apricot utilise le mécanisme d'IPC Message Queue : Le programme cgi envoie effectivement des commandes dans une file de message mqueue_t que le serveur va lire.
L'interface d'administration du serveur est accessible via le programme cgi admin et permet de visualiser le fichier journal, de l'effacer, de redémarrer le serveur et de redimensionner la pool de workers à une taille inférieure.
Il est donc facile d'ajouter ou de supprimer des commandes d'administration à Apricot.

4 Gestion dynamique du pool de processus (**)

La gestion dynamique du pool de processus se fait avec un tableau de workers qui est alloué/réalloué dynamiquement.
Notre implémentation est assez robuste aux erreur d'allocations lors d'un potentiel redimensionnement de la pool.
Par ailleurs, lors d'une diminution de taille, si un worker doit être arrêté, il fini de servir le client si nécéssaire.

5 Gestion de cookies (version 2) (***)

La gestion des cookies se fait grâce à un identifiant de cookie (cookie_id) qui référence un fichier côté serveur qui contient toutes les données associées par les programmes cgi.

Lors de l'arrêt du serveur, les fichiers cookies sont effacés.
Quand un fichier cookie côté serveur est supprimé, un nouveau cookie est généré.
Le programme cgi cookie permet de tester cette fonctionnalité de cookies : il mémorise avec un cookie le contenu d'une variable message passée en paramètre GET au programme précédement, il compte également les visites d'un même visiteur et affiche le contenu d'une chaine stockée dans un cookie contenant un caractère de retour à la ligne. On vérifie ainsi que l'encodage des données stockées dans le cookie est correct.

Afin de stocker pour chaque cookie un ensemble de paires (clé, valeur), on implémente des arbres à préfixe qui garantissent un accès à la valeur de complexité proportionnelle à la longueur de la clé.

6 Programme CGI complexe (**)

Le programme CGI complexe consiste en un appel à exec en pensant à bien attendre le fils créé et de flush stdout après avoir écrit le "content-type".

On appelle le programme traceroute, disponible depuis l'interface d'administration.

7 Méthode POST (**)

Lors du transfer de données depuis le client via un formulaire, on redirige sur l'entrée standard les données, ainsi le cgi à juste à lire les données depuis l'entrée standard.
Nous avons implémenté un "wrapper" qui nous facilite la lecture des données POST et GET, et la gestion des cookies.

8 Verrouillage

Pour assurer la cohérence des fichiers, des cookies notamment, seule une instance du serveur ne peut être exécutée en même temps, cette fonctionnalité est implémentée par verrouillage de fichier.

9 Démon

Afin de pouvoir faire fonctionner le serveur en arrière-plan et après déconnexion de l'utilisateur, le serveur se lance comme un démon par défaut. Pour désactiver cette fonctionnalité, on peut utiliser l'option d.