La technologie mod_perl

Linux Expo 2001

Benjamin Drieu – Benjamin.Drieu@alcove.fr

02. Février 2001

Plan

- Introduction
- mod_perl
- Comparaisons
- Conclusion

Plan détaillé

- Introduction
  - Perl
  - Apache
  - Contenu dynamique et CGI

Plan détaillé (2)

- mod_perl
  - Principes
  - Fonctionnalités
  - Problèmes, solutions

Plan détaillé (3)

- Comparaisons
  - mod_perl vs. CGI
  - mod_perl vs. php3

Plan détaillé (3)

- Conclusion
  - *Success stories*
  - Liens utiles

Perl

- Langage de script
- Langage pré-compilé (bytecode)
- Développé par Larry Wall
- Héritage de awk, sed et shell

Perl (2)

- Buts de Perl
  - traiter des flux arbitraires de texte
  - faciliter les tâches d'administration système
  - être efficace (facile à utiliser, efficace, exhaustif)

Perl 5

- Depuis Perl 5 (réécriture complète) :
  - syntaxe simplifiée
  - programmation orientée-objet
  - abstraction des bases de données (DBI)
  - auto-chargement de routines

CPAN

- Archive de modules disponibles
- Résoudre profusion anarchique de modules
- Création du CPAN en 1996
- Solution comparable au CTAN

CPAN (2)

- Exemples de modules du CPAN
  - graphisme: Gtk, OpenGL, GD
  - réseau: Net::SMTP, Net::LDAP, Net::Whois
  - base de données: DBI/DBD, DB/DBM

Perl et le web

- La programmation du web fait intervenir principalement du traitement de chaînes
- Développement avec Perl très rapide
- Module CGI par Lincoln Stein
- Abstraction DBI
- Supporte la comparaison avec des programmes compilés (C,C++)
- Alternatives usuelles: php, python, jserv

Apache

- Le serveur web le plus utilisé (60% en décembre 2000)
- Porté sur virtuellement tous les UNIX et sur MS-Windows
- Bonnes performances
- Nombreuses fonctionnalités (authentification, contrôle d'accès, proxy, bases de données, ...)

Les modules d'Apache

- Architecture modulaire
  - Écriture en C
  - Chargement dynamique
  - Principe du handler

Le contenu dynamique

- Exécution d'un programme
- Retour de la sortie standard au client HTTP
- Plusieurs stratégies
  - exécuter un processus externe au serveur (CGI)
  - exécuter un programme interprété par le serveur Apache (php, mod_perl, ssi)
  - définir un handler dédié (mod_status)

La CGI

- La solution traditionnelle pour le contenu dynamique
- Exécution d'un processus externe
- Envoi du résultat au client
- Indépendance vis-à-vis du langage
- Indépendance de la mémoire

Problèmes de la CGI

- Interaction très limitée
- Temps de démarrage d'un processus assez coûteux
- Pas de mémoire persistante

mod_perl

- mod_perl résout ces problèmes en incorporant au serveur Apache un interpréteur Perl
- Perl évalue dynamiquement du code
- mod_perl fournit une API Apache complète en Perl

mod_perl (2)

- Un avantage immédiat: l'exécution de scripts CGI Perl (*Apache::Registry*)
- L'interpréteur Perl est
  - chargé au démarrage
  - partagé par tous les processus Apache
  - utilisé à chaque requête pour évaluer du code Perl
- Un script chargé par l'interpréteur Perl est conservé en mémoire

Script CGI classique

```
use strict;

print "Content-type: text/plain\n\n";

my $counter = 0;

for (1..5) {
  $counter++;
  print $counter;
}
```

Avec mod_perl

```
package Apache::ROOT::perl::conference::compteur_2epl;
  use Apache qw(exit);
  sub handler {
    use strict;

    print "Content-type: text/plain\n\n";

    my $counter = 0;

    for (1..5) {
      $counter++;
      print $counter;
    }
  }
```

Apache::Registry

- Mise à jour du script si le fichier est plus récent que la version compilée.
- Problème: pas de destruction du processus à la fin de l'exécution (effets de bords possibles).
- Solution peu coûteuse: *Apache::Perlrun* recompile le script à chaque invocation et émule l'environnement CGI.

Perl dans un document HTML

- Possibilité d'incorporer du Perl dans un document HTML (comme pour PHP)
  - HTML::Embperl
  - HTML::Masson
  - Apache::ASP
  - AxKit

Développer un handler

- mod_perl peut intervenir à n'importe quelle étape de la vie du serveur
- Développement d'un module avec Perl tout comme avec le C
- Possibilité d'interfacer Apache avec n'importe quel module Perl

Quelques exemples d'utilisation

- Configuration (`httpd.conf`) entièrement en Perl
- Stockage des sessions (via DBI, en mémoire, dans un fichier)
- Sauvegarde des logs dans une base de données
- Exécution de code Perl en cas de réception de signaux
- Monitoring des processus Apache

La mémoire, goulet d'étranglement

- Utiliser un deuxième serveur pour gérer les pages statiques
- Charger des modules au démarrage
- Charger les scripts au démarrage
- Contrôler la taille des processus

Comparaisons

- mod_perl vs. CGI
- mod_perl vs. php3

Apache::Registry vs. CGI

- Performance multipliée selon les cas par un facteur de 4 à 20
- Ajout de fonctionnalités (connexions persistantes à une base de données, handlers)
- Problèmatique du portage de l'existant

mod_perl vs. php

- Performances comparables
- Buts différents:
  - php orienté programmation web
  - mod_perl plus large (API d'Apache)
- Développement d'applications simples plus rapide en php

mod_perl aujourd'hui

- 1.800.000 sites utilisant mod_perl en décembre 2000 (360.000 en janvier 2000)
- utilisé sur 9% des serveurs Apache
- et pourtant peu de développeurs mod_perl

Success stories

- CPAN
- slashdot.org
- eToys.com
- PBS online

Conclusion

- une technologie puissante mais complexe
- mod_perl, pour faire quoi ?

Liens

- http://www.apache.org/(`http://www.apache.org/`)
- http://www.perl.com/(`http://www.perl.com/`)
- http://perl.apache.org/(`http://perl.apache.org/`)
- http://take23.org/(`http://take23.org/`)