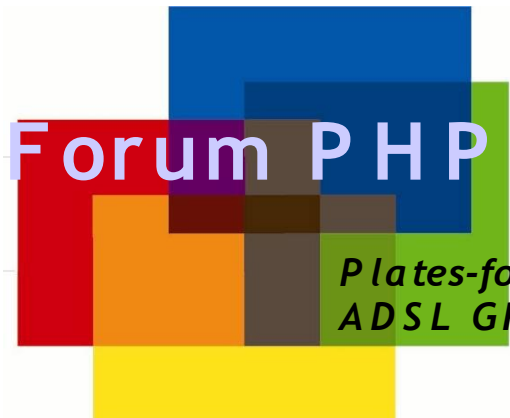


Auditer et optimiser vos applications PHP

Par Franck LOMBARDI, Cyril PIERRE de GEYER, Guillaume PONCON



Plates-formes de service
ADSL GROUPE CANAL +

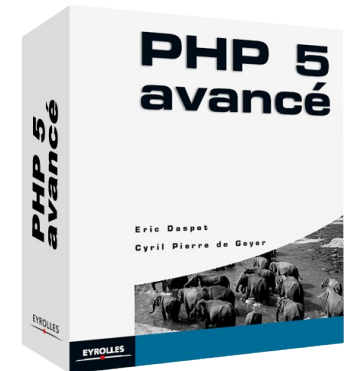
Présentation



Franck LOMBARDI
Canalsat, Ingénieur intégrateur

Cyril PIERRE de GEYER,
Anaska, auteur du livre PHP 5 avancé

Guillaume PONCON
Anaska, auteur du livre Best Practices PHP 5



Combien d'entre vous...

- Utilisent un débogueur ?
- Utilisent un profiler ?
- Utilisent un cache d'op-codes ?
- Déploient leur application sur plusieurs serveurs ?

L'offre « audit et performances »

Problématique :

Un projet qui devient une application critique.

Comment le fiabiliser ?

Objectifs :

Faciliter l'évolution fonctionnelle des applications.

Optimiser les performances et la scalabilité.

Valider la sécurité de vos plateformes.

Faciliter la maintenance.



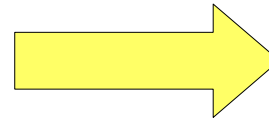
Nos partenaires sur
ce type de mission

Zend

L'offre « audit et performances »

Les différents aspects audités :

- La qualité du code PHP existant
- Les appels aux bases de données
- Les environnements de
 - Développement
 - Préproduction
 - Production
- Les problèmes de performances
- La sécurité
- Les procédures de maintenance.



Rapport d'analyse
Planification
Recommandations

Auditer et optimiser vos applications PHP

➔ A. Les outils pour auditer vos applications

1. Profiler vos applications
2. Déboguer vos applications
3. XDebug + KCacheGrind
4. Zend Platform

B: Optimiser vos applications

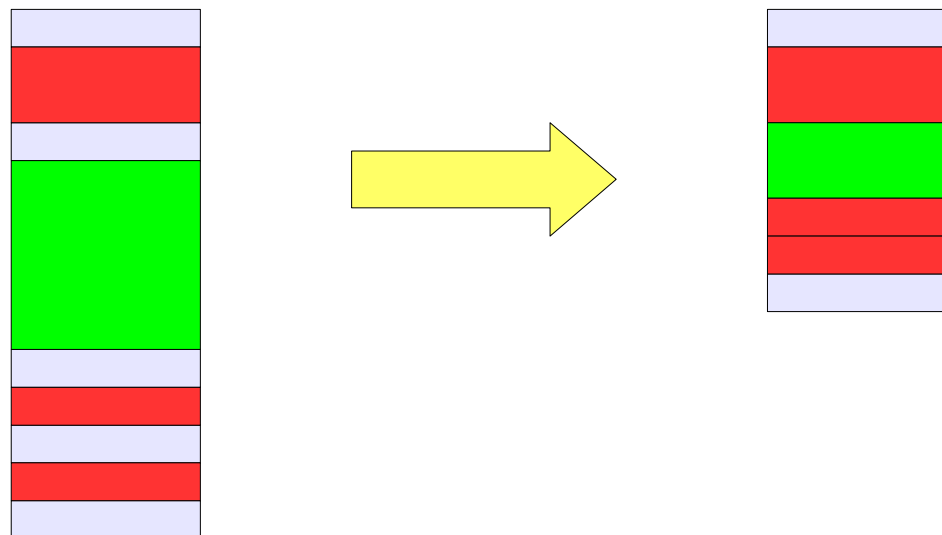
5. Optimiser l'architecture PHP
6. Optimiser l'architecture MySQL
7. Optimiser le code : tips
8. Tordons le cou aux mythes

C: Retour d'experience : Canal Satellite

9. Le contexte
10. La première mise en oeuvre
11. Architecture après refonte

Profiler vos applications

- A quoi sert le profiling ?
 - Repérer les opérations coûteuses
 - Repérer les doublons



Profiler vos applications

- Principe du profiling
 - Capturer et enregistrer les instructions élémentaires
 - Les afficher et les manipuler avec un logiciel adapté



Profiler vos applications

Quelques outils

- APD (Profiler)
- XDebug (Profiler & Debugger)
- DBG (Profiler & Debugger)
- Zend Platform (Profiler & Debugger)
- KCacheGrind (Visualisation de profils)
- WinCacheGrind (Visualisation de profils)

Déboguer vos applications

Et pourquoi déboguer ?

- Analyser le comportement d'un algorithme
- Tenter de résoudre un problème spécifique
- Executer une requête avec des paramètres spécifiques

Déboguer vos applications

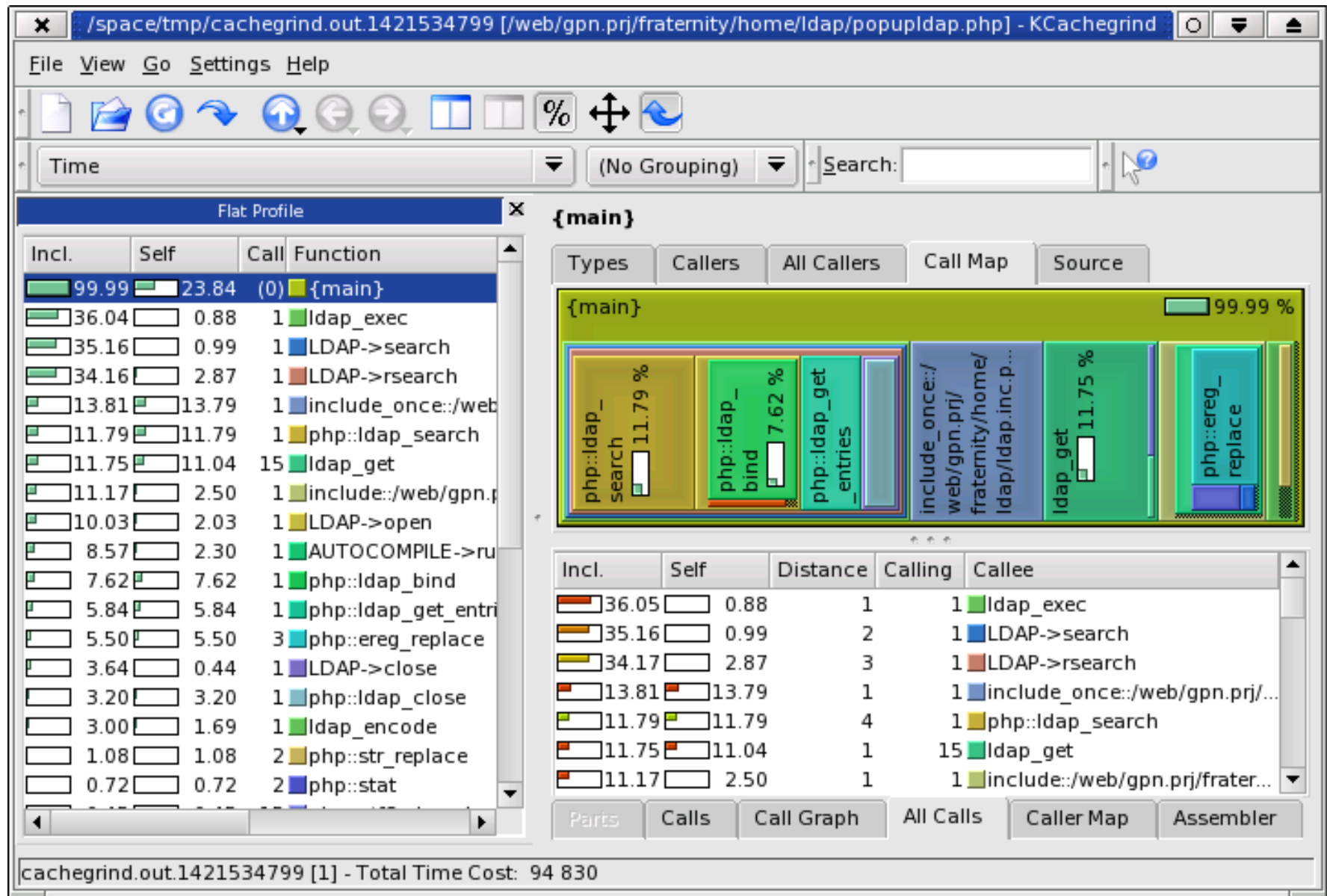
Principe du débogage

- Permettre l'exécution du code pas à pas
- Permettre l'exécution du code jusqu'à un certain point (point d'arrêt)
- Modifier le contexte en cours d'exécution (changer la valeur d'une variable, modifier l'environnement, etc.)

Déboguer vos applications

Quelques outils

- XDebug, DBG
- PHP Eclipse + DBG
- Zend Studio + Zend Platform



Démo : XDebug + KCacheGrind

- Exécution d'une requête
- Visualisation du profil en mode texte
- Visualisation du profil avec KCacheGrind

Démo : Zend Studio + Zend Platform

- Débogage avec l'interpréteur interne de Zend Studio
- Débogage avec la Zend Platform
- Profiling

Auditer et optimiser vos applications PHP

A. Les outils pour auditer vos applications

1. Profiler vos applications
2. Déboguer vos applications
3. XDebug + KCacheGrind
4. Zend Platform

➡ B. Optimiser vos applications

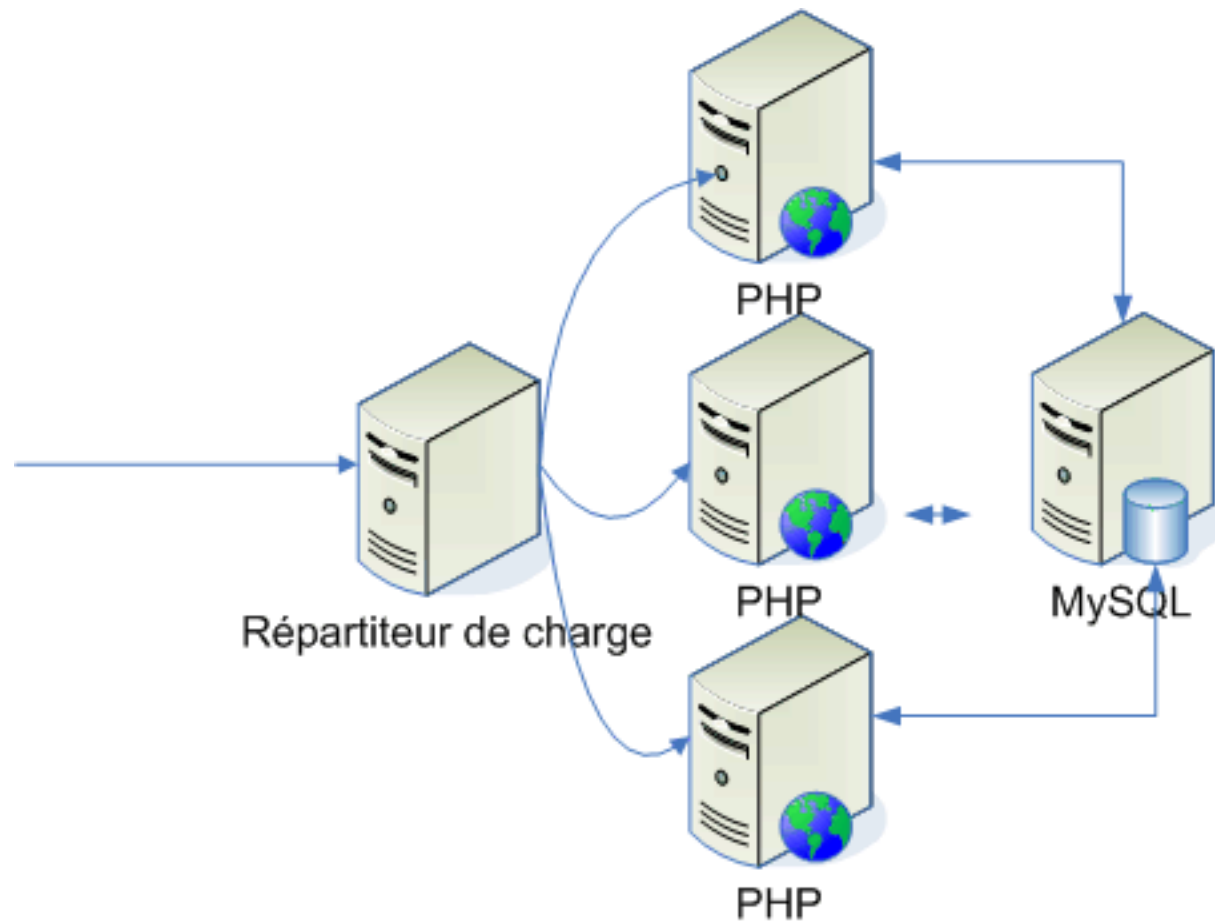
5. Optimiser l'architecture PHP
6. Optimiser l'architecture MySQL
7. Optimiser le code : tips
8. Tordons le cou aux mythes

C. Retour d'expérience : Canal Satellite

9. Le contexte
10. La première mise en oeuvre
11. Architecture après refonte

Optimiser PHP / Architecture

(Répartiteur de charge)



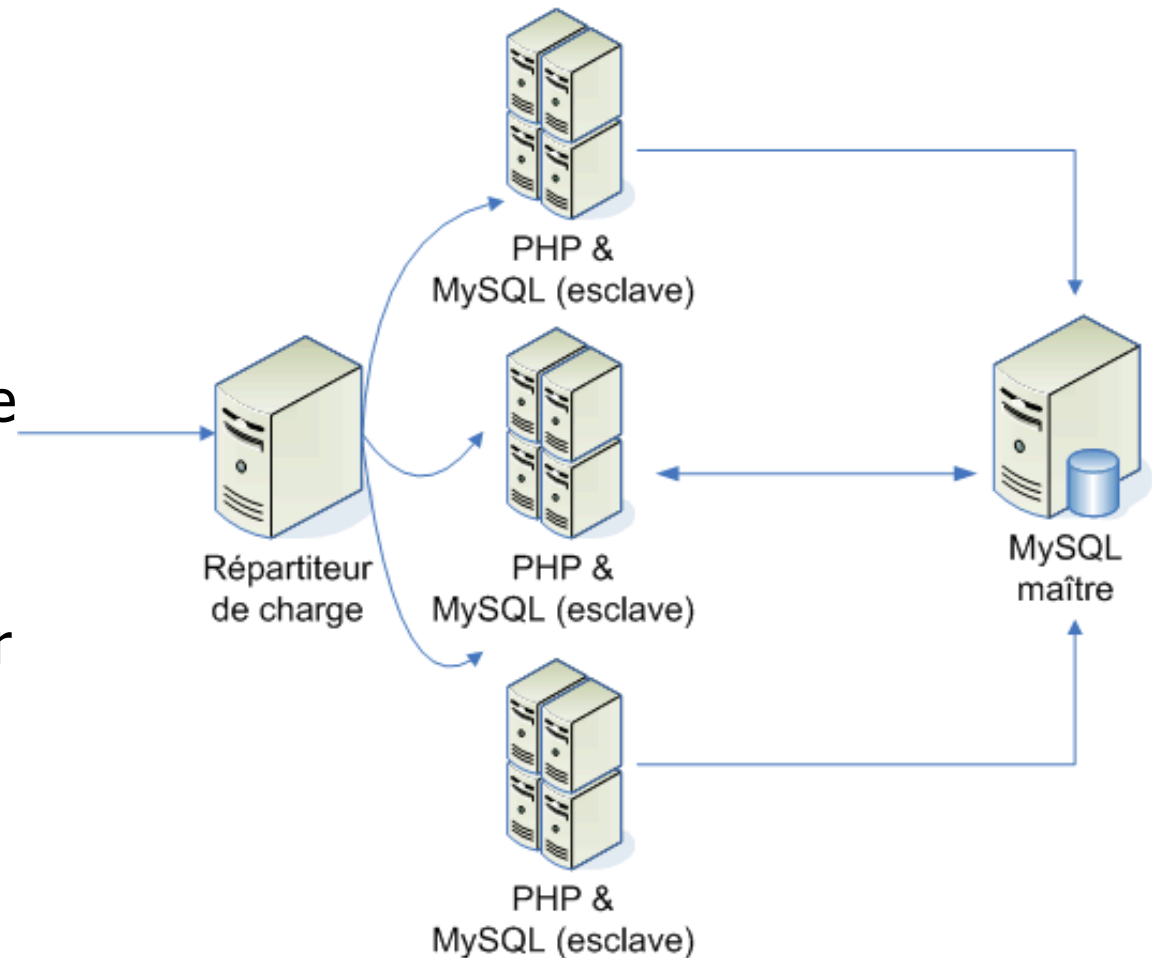
Optimiser MySQL / Architecture

(Réplication MySQL)

Solution classique

Avantage : évite de coder le choix du serveur de manière aléatoire.

Désavantage : data lag (gap de temps entre l'écriture sur le maître et la maj sur les esclaves).



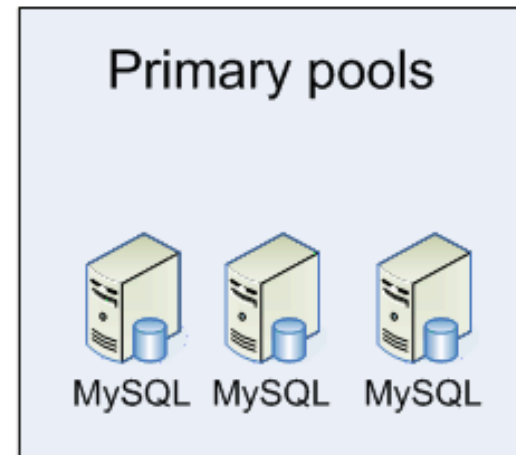
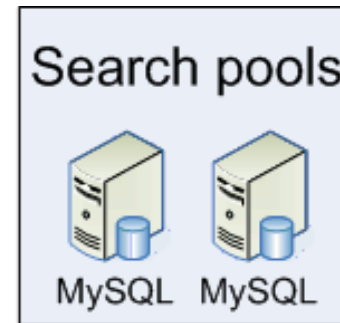
Optimiser MySQL / Architecture

(Réplication MySQL)

Des esclaves spécialisés

Permet d'avoir un accès rapide sur les données prioritaires.

Implique plus de travail dans le code.

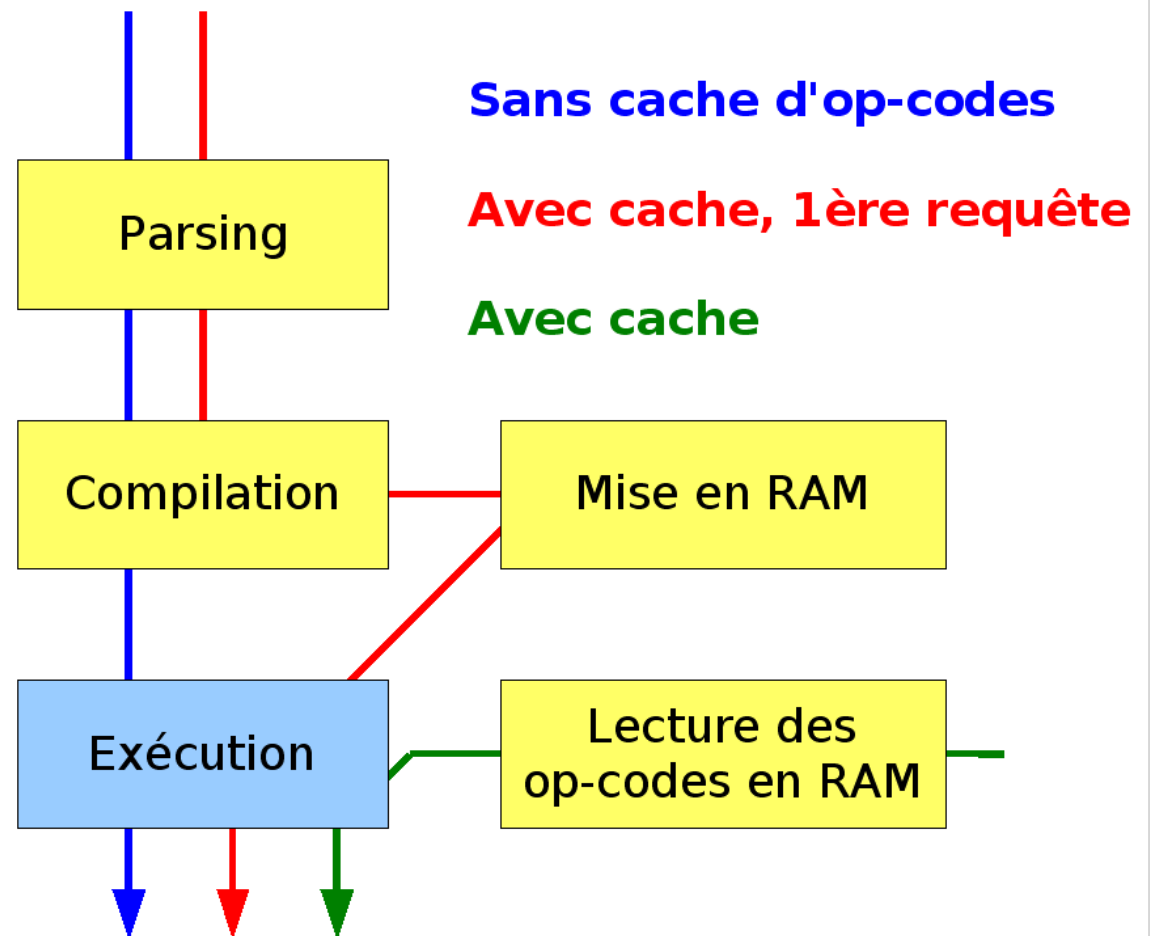


Optimiser PHP / Architecture

(Cache d'OpCode)

chaque script est compilé une seule fois pour chaque révision.

on diminue les flux de lecture écriture fichier. L'Opcode est lu directement dans la mémoire plutôt de d'être parsé sur le disque.



Optimiser PHP / Architecture

(Cache d'OpCode)

Les outils existants

APC

e-accelerator

PHP Accelerator (ion cube)

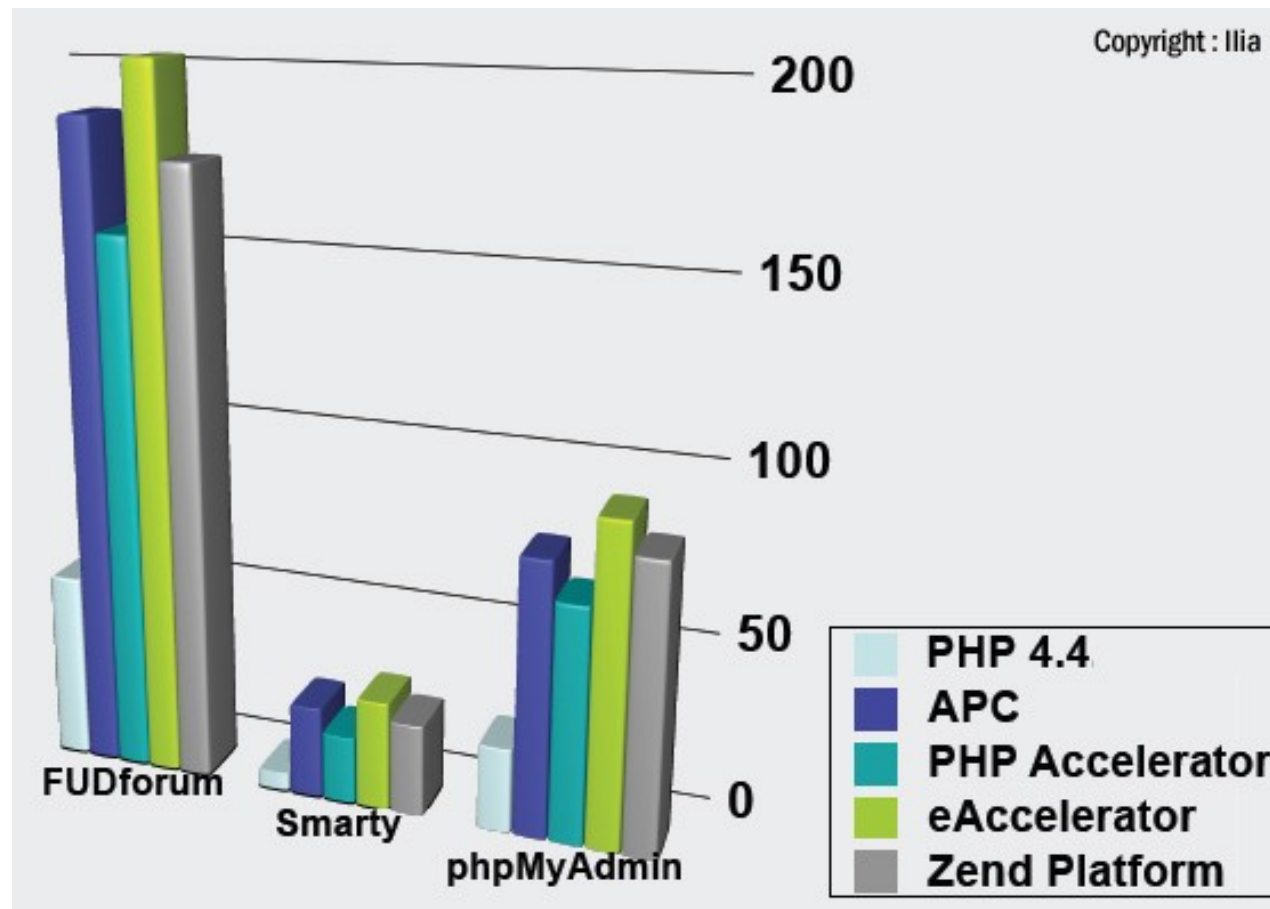
ZendPlatform



Avec PHP 6 APC sera intégré nativement àPHP

Optimiser PHP / Architecture

Comparatif de performances (**Cache d'OpCode**)



Optimiser PHP / Architecture

(Cache de contenu)

Utilisez un cache de contenu quand c'est possible

Une page qui n'a pas besoin d'être générée à chaque fois doit être prégénérée.

Plutôt que de prégénérer le contenu à intervalles réguliers on peut le faire à la demande. Ex : Jpcache.

Optimiser PHP / Architecture (Compression de contenu)

Moins de sorties == Plus de rapidité

- La bande passante coûte cher
- Les pages s'affichent plus vite chez les clients
- La bande passante == goulot d'étranglement



Extension apache : mod_gzip

Optimiser PHP / Architecture

Utilisez des serveurs de média

Apache est parfait pour les requêtes dynamiques mais pour les requêtes statiques d'autres serveurs sont plus appropriés :

- lighthttpd
- Boa
- Tux
- thttpd



Amélioration des temps de réponses de l'ordre de 300 %

Un des pbms d'apache c'est que le nombre de process est limité et qu'il ne rend la main qu'une fois son action terminée. Donc par exemple il peut se bloquer si on télécharge 200 fichiers vidéo de 2go. Il ne répondra plus à rien.

Optimisation : tips

Il est important de disposer de disques dur rapides et de beaucoup de mémoire vive.

Il est recommandé de limiter le nombre d'accès aux fichiers.

Il faut utiliser la POO avec modération car son utilisation peut parfois engendrer des pertes de performances par rapport à un modèle procédural.

N'utilisez pas l'opérateur @

```
@action();  
$old = ini_set("error_reporting", 0);  
action();  
ini_set("error_reporting", $old);
```

Auditer et optimiser vos applications PHP

A. Les outils pour auditer vos applications

1. Profiler vos applications
2. Déboguer vos applications
3. XDebug + KCacheGrind
4. Zend Platform

B. Optimiser vos applications

5. Optimiser l'architecture PHP
6. Optimiser l'architecture MySQL
7. Optimiser le code : tips
8. Tordons le cou aux mythes



C. Retour d'expérience : Canal Satellite

9. Le contexte
10. La première mise en oeuvre
11. Architecture après refonte

Interactivité CANALSAT

Services dits « opérateur », faisant partie intégrante des Offres ADSL GROUPE CANAL+ :



Zapping



Mosaïque



Pilote



Menu



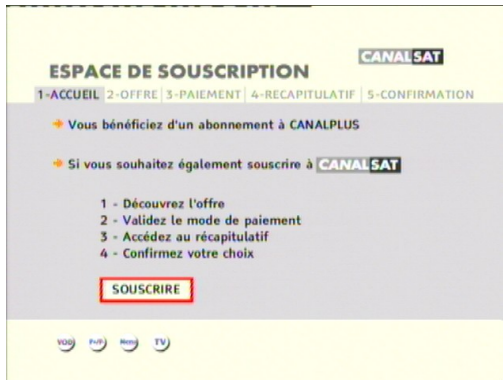
Zapliste



Guide

Interactivité CANALSAT

Services de type « self care »



Espace de souscription TVi

Services spécifiques



Service de PPV FOOT

Service de VOD

Contexte

CANALSAT sur l'ADSL

Décodeurs pauvres (Navigateur Web embarqué)

Technologie Web pour une problématique de TV interactive

Utilisation de LAPP (Linux, Apache, Postgres, PHP)

Puis utilisation de LAMP (Linux, Apache, MySQL, PHP)

Première approche

Situation :

- Exploration d'une nouvelle niche
- 1 seul FAI partenaire
- Peu d'abonnés cibles
- Impératifs de démarrage à court terme (avant la concurrence)

Conséquences :

- Aucun retour d'expérience pour évaluer le besoin
- Parc de décodeurs homogènes -> 1 seul modèle de navigateur
- Des délais courts

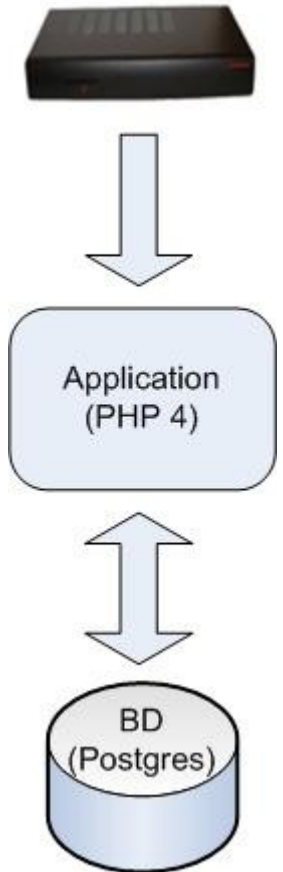
Première approche

Choix techniques :

- Problématique proche du Web -> Technologie Web
- Logiciels libres :
 - Système GNU/Linux très utilisé en interne
 - LAMP et LAPP -> des classiques du Web
 - De nombreuses sources d'informations disponibles

Résultats :

- Une application développée rapidement
- Une application fortement liée aux caractéristiques des décodeurs
- Une application ne proposant que les services dits « opérateur »



Seconde étape de développement

Nouveaux besoins :

- 6 principaux FAI partenaires
- De plus en plus d'abonnés
- Nouveaux services spécifiques aux FAI
- Nouveaux services en exclusivité avec certains FAI
 - PPV FOOT
 - VOD
- Impératifs de démarrage à court terme (avant la concurrence)

Conséquences :

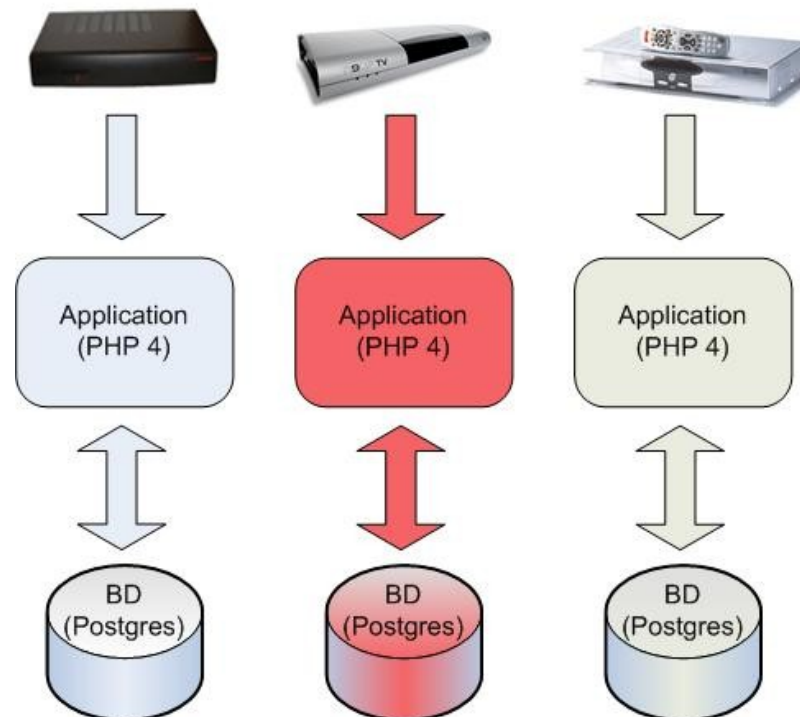
- Parc de décodeurs hétérogènes -> différents navigateurs
- Des services différents suivant le décodeurs
- Des délais courts

Seconde étape de développement

Choix techniques :

Duplication de l'application pour chaque type de décodeurs

Développement de services uniquement sur certaines versions de l'application



Seconde étape de développement

Résultats :

- Plusieurs versions de l'application à maintenir
- Des versions intégrant de + en + de spécificités
- Difficultés d'intégration des nouveaux services
- Données dupliquées
- ...
- Coût des évolutions et des corrections multiplié par le nombre de versions de l'application

Industrialisation

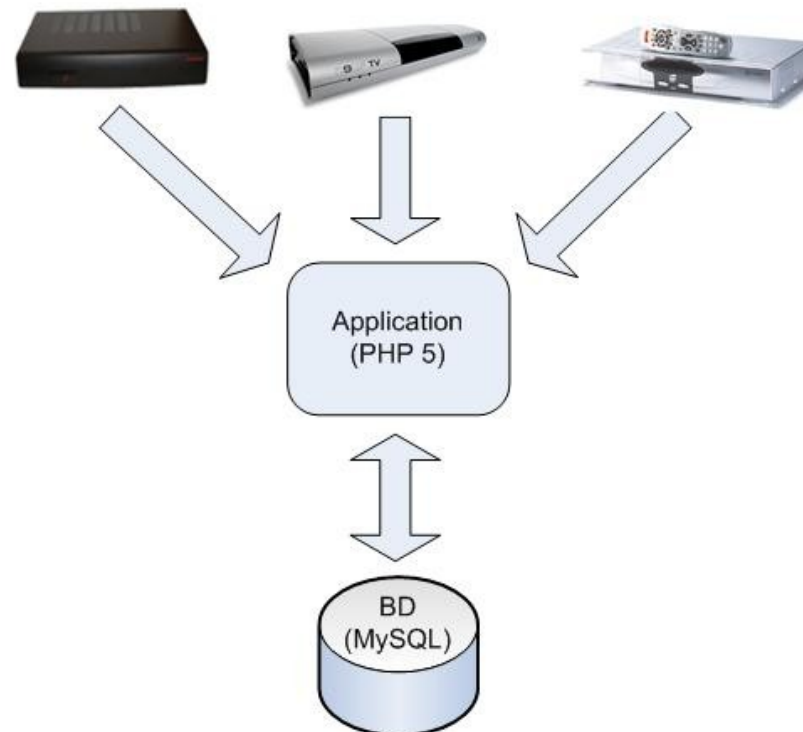
Un besoin de rationalisation émergeant :

- Application unique, modulaire et multi-décodeurs
 - De nombreux services communs
 - Des données communes
 - Des scénarios d'utilisations communs
 - Des évolutions et corrections communes
 - ...
-
- Étude de faisabilité : audit ANASKA sur la plateforme actuelle et sur les choix futurs

Industrialisation

Résultats :

- Modèle MVC (Modèle – Vue – Contrôleur)
- Mise à jours de nos plate-formes
- Choix de nouveaux outils et bibliothèques plus adaptés (PHP5, MySQL...)



Questions ?



**Besoin d'information sur l'offre
« audit et performances »**

romain@anaska.com
ou rdv sur notre stand

Nos formations PHP de niveau 3 — PHP Best Practices

PHP Tuning

