PHP & performances

Forum PHP Paris 2005

Éric Daspet

Plan de la session

- Avant-propos
- Configuration
- Code PHP
- Protocole HTTP
- Récapitulatif

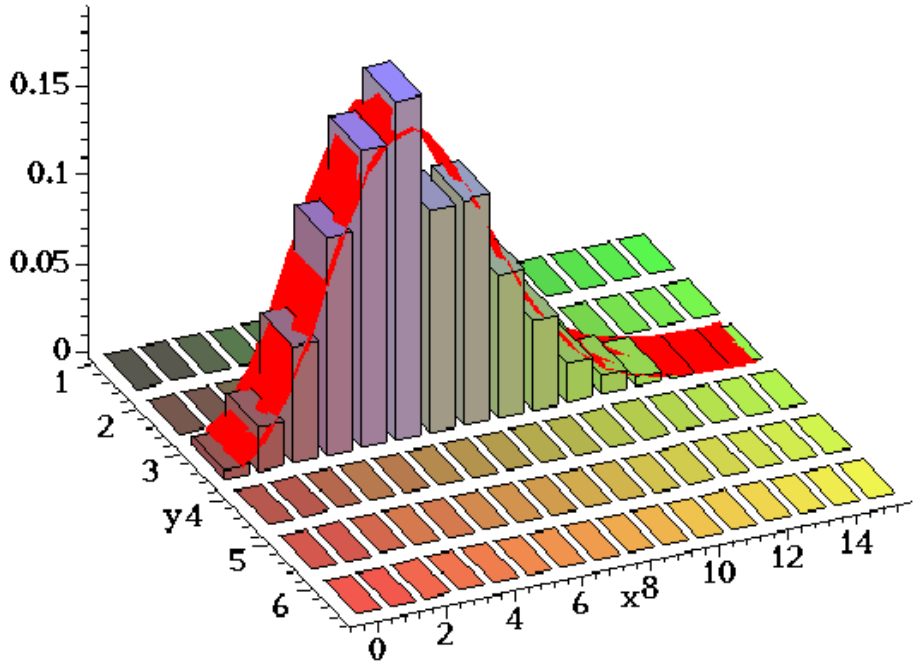
Quelle(s) performance(s) ?

- **Logiciel** : Serveur Web, PHP, SGBD
  - Nécessite souvent de l'expertise
- **Matériel** : processeur, mémoire, disque, réseau
  - Simple à ajouter
- **Utilisateur** : débit et réactivité
  - « Ressenti » difficile à mesurer

La problématique

- Et moi ?
  - Quel facteur limitatif ?
  - Que sacrifier en retour ?
  - Comment y remédier ?
  - La performance est-elle ma priorité ?

Il n'y aura (presque) pas de chiffres



- Peu fiables
- Dépendent :
  - De la configuration
  - De la charge système
  - De votre matériel
  - De vos besoins
  - De votre applicatif
  - etc.

Plan de la session

- Avant-propos
- Configuration
- Code PHP
- Protocole HTTP
- Récapitulatif

Mauvaises rumeurs

- Recompiler PHP et Apache depuis les sources
  - Des experts l'ont déjà fait pour vous
  - Utile uniquement pour des besoins spécifiques
  - Optimisations de compilation négligeables
- Compilation statique (opposé à dynamique)
  - Moins souple (module non indépendants)
  - Aucune différence en module apache ou FastCGI

Différents modes d'accès

- CGI : utilisé pour suexec
- Module Apache : utilisé pour les performances
- **FastCGI** : répond à tout ou presque, peu utilisé

	CGI	Module Apache	FastCGI
Processus indépendants ?	**Oui**	Non	**Oui**
Serveurs indépendants ?	Non	Non	**Oui**
Configuration par application ?	Non	**Oui**	**Oui**
Performant ?	Non	**Oui**	**Oui**
Configuration / optimisation ?	Non	Commune Apache	**Oui, par appli.**
Utilisateurs indépendants ?	**Oui**	Non	**Oui**

Quel serveur Web ?

- Apache
  - Éprouvé
  - Modules nombreux
- **Lighttpd**
  - Simple
  - **Rapide (fichiers statiques)**
  - **Peu gourmand**
  - http://lighttpd.net/

Caches d'opcode

- Pourquoi ?
  - Une compilation, plusieurs exécutions
  - Optimisation du code
- Les noms à retenir : APC, e-accelerator, Zend
- Les résultats ?
  - Calcul pur : faibles ou inexistants
  - Applications réelles : **gains importants (x2)**

Quelle configuration Apache ?

- Apache
  - Limitez les override (.htaccess)
  - Limitez la profondeur des répertoires
  - Limitez DirectoryIndex
  - Pas de négociation de contenu
  - Pas de rewrite
  - Autorisez les liens symboliques
  - Désactivez les journaux inutiles
  - Désactivez la résolution DNS (n'utilisez que des IP)

Quelle configuration PHP ?

- Utiliser les limitations propres à PHP si besoin
  - max_input_time, max_execution_time, memory_limit
- Retirer l'inutile
  - register_long_arrays ($HTTP_*_VARS)
  - register_argc_argv
  - always_populate_raw_post_data
  - magic_quotes_*
  - include_path (à limiter au strict nécessaire)
  - open_basedir (à limiter ou retirer)

Quelle configuration PHP ?

- Les sessions
  - Régler la fréquence du ramasse-miette
    - Trop lent = gros ralentissement peu fréquent
    - Trop rapide = petit ralentissement mais très fréquent
  - Désactiver la réécriture d'URL (session.trans_sid)
  - Désactiver le démarrage automatique
  - Changer le système de stockage (files => mm)
    - ou utiliser un système de fichier en mémoire
  - Profiter de session.save_path="N;/path"

Réglage des tampons de sortie ?

- Négatif ou positif, testez *votre* cas
- Diminue le nombre d'I/O système

- Dans PHP :
  - output_buffering (valeur true ou 4096)
- Dans Apache :
  - SendBufferSize (valeur PageSize)

Multiplier les serveurs

- N'hésitez pas à avoir plusieurs serveurs
- Chaque requête est indépendante
  - Donc facilement parallélisable
  - Aucune configuration spécifique
- Solutions :
  - DNS round-robin
  - Répartiteur TCP/IP
  - Répartiteur HTTP

Plan de la session

- Avant-propos
- Configuration
- Code PHP
- Protocole HTTP
- Récapitulatif

Utiliser les fonctions natives

- PHP est lent ... par rapport au C
  - Les fonctions proposées par PHP sont codées en C
  - PHP en a des milliers : ne réinventez pas la roue
- Exemples :
  - file_get_contents(), file_put_contents(), et glob()
  - strncmp(), http_build_query()
- PECL a ce qui n'est pas standard
  - Classes de gestion HTTP, systèmes de paiement, ...

Utiliser les fonctions natives

- Rien ne remplit vos besoins ?
  - Utilisez les bibliothèques de votre système
  - Créez votre extension PHP avec vos fonctions
  - Tout cela est très simple
- PHP sert de colle

  - entre le système, les données, les bibliothèques de traitement, et l'affichage
  - ... laissez le reste aux extensions

Astuces de syntaxe

- Références comme des raccourcis ==>
- Limitez l'accès aux sessions ou fermez-les dès que possible
- Passez vous des regexp quand vous le pouvez
- Sinon n'utilisez pas ereg_* mais preg_*

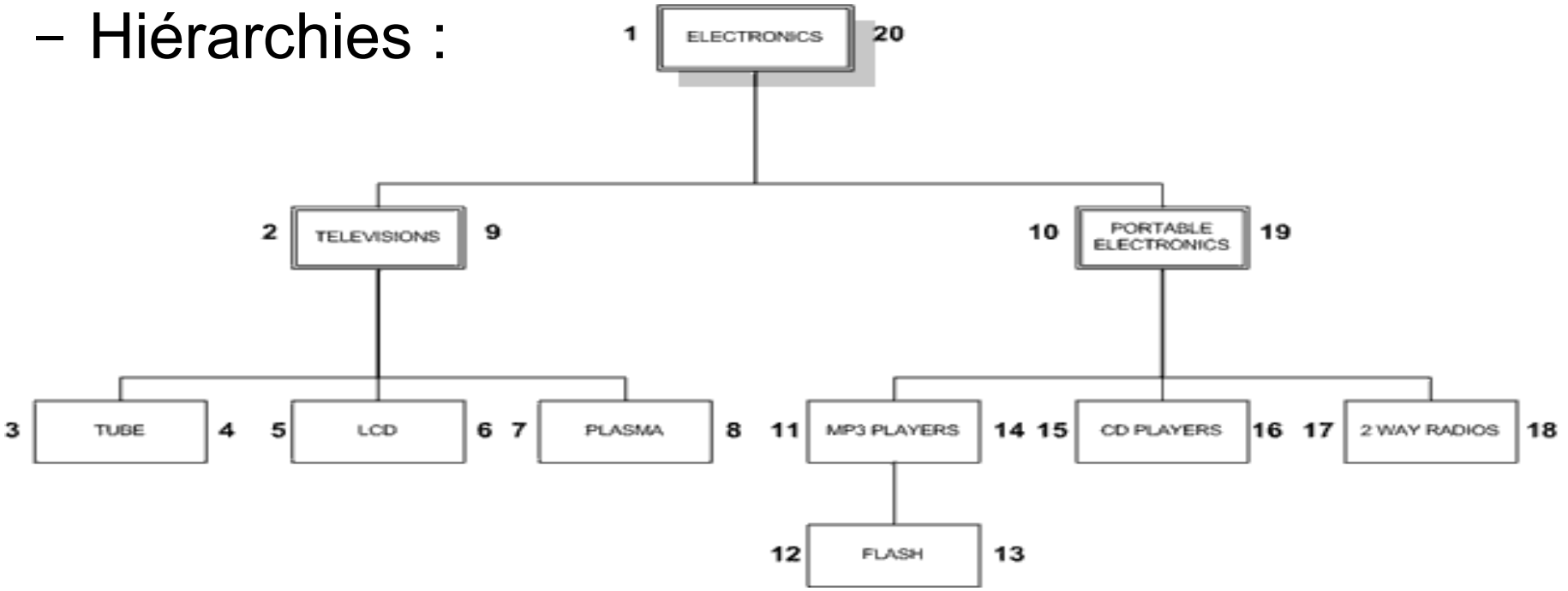
```
for($i=12; $i<56; $i++)
{
echo $tab['a']['test'][45][$i];
}
--------------------------------

$tmp =& $tab['a']['test'][45] ;
for($i=12; $i<56; $i++)
{
    echo $tmp[$i] ;
}
```

Mauvaises idées

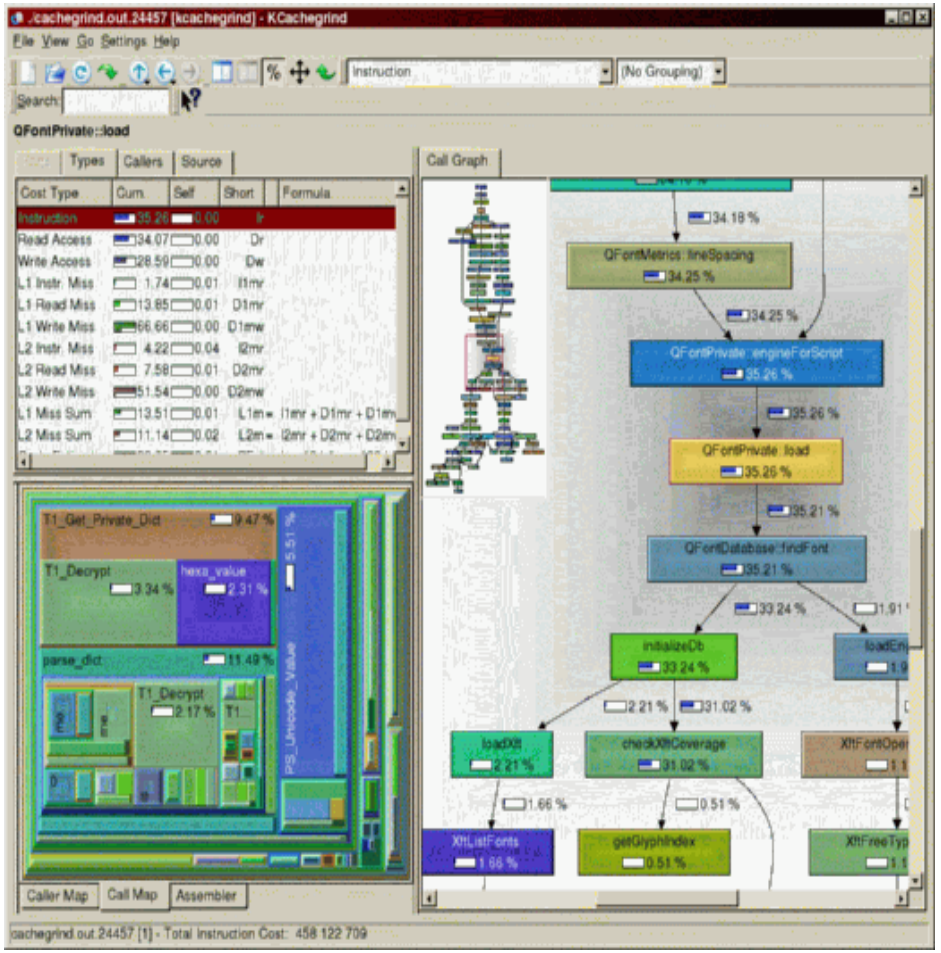
- Ne vous fiez pas aux rumeurs d'Internet
  - Gains négligeables en performances
    - "toto" <-> 'toto'
  - Certaines croyances sont fausses
    - CONSTANTE <-> $variable
  - Pertes importantes en lisibilité / maintenance
    - if () { ... } else { ... } <-> ()?(...):(...)
- Ne vous occupez pas trop de la syntaxe
  - Vous avez plus à y perdre qu'à y gagner

Accès aux données et SGBDR

- Limitez l'accès au SGBD
  - Les fichiers sont souvent plus pertinents
- Optimisez les accès au SGBDR
  - Requêtes mal codées et index manquants
  - Hiérarchies :



Profilage, savoir quoi optimiser



- La grosse perte de performances n'est pas forcément là où vous croyez
- Vos armes :
  - Xdebug
  - Kcachegrind

Utilisez des caches

- Ne pas refaire un traitement plusieurs fois
  - Votre contenu est-il vraiment dynamique ?
- Solutions :
  - JPCache et Pear::Cache_Lite
  - Cache de requêtes Mysql 4
- **Penser à la pré-génération**
  - Créer des fichiers statiques à la demande
  - Utiliser la redirection des erreurs 404
  - Effacer régulièrement les vieux contenus

Plan de la session

- Avant-propos
- Configuration
- Code PHP
- Protocole HTTP
- Récapitulatif

Utiliser les proxy

- Idée : le proxy appelle une fois la page et peut la servir à plusieurs personnes
  - On déclare si une page est publique ou privée
    - header("Cache-Control: public, max-age=3600");
    - header("Cache-Control: no-cache, must-revalidate");
  - Plusieurs clients d'une même entreprise = une seule requête
- Attention, avec les sessions PHP déclare automatiquement des pages « privées »

Cache du navigateur

- Idée : ne pas re-télécharger un contenu qui n'a pas changé depuis la dernière fois
  - client : « je veux le contenu s'il a été modifiée »
  - serveur : « il n'a pas changé, « code 304 »
  - client : « ok, je reprends ma copie en cache »
- Avec PHP :
  - $_SERVER['IF_MODIFIED_SINCE']
  - $_SERVER["HTTP_IF_NONE_MATCH"]

Cache du navigateur

```php
// on récupère et envoie la date de dernière modification
$modif = gmdate('D, d M Y H:i:s', filemtime($cache_file)) ;
header("Last-Modified: $modif GMT");

// on vérifie si le contenu a changé
$if=substr(@$_SERVER['HTTP_IF_MODIFIED_SINCE'],0,29) ;
if ($date!='' && strtotime($if)>=filemtime($cache_file)) {
    header('Not Modified', TRUE, 304);
}

// S'il a changé et si c'est nécessaire, on renvoie le contenu
elseif ($_SERVER['REQUEST_METHOD'] != 'HEAD') {
    readfile($cache_file) ;
}
```

Compression

- Idée : transmettre le contenu compressé (zip)
  - Réduction du temps d'attente du client
  - Gain financier important sur la bande passante
  - Taille divisée par 5 à 10
  - On utilise 5% de proc pour diviser par 5 le réseau
- Moyen
  - Sous Apache : mod_gzip, mod_deflate
  - Sous PHP : zlib.output_compression=On

Durée de vie

- Idée : ne pas télécharger trop souvent les graphiques ou contenus qui ont une durée de vie connue à l'avance
- Moyen : entête HTTP « Expires »
  - Le navigateur reprend sa copie en cache
  - Il ne fait aucune requête au serveur Web
- En PHP :
  - header("Expires:Mon, 25 Nov 2005 12:00:00 GMT");
  - Calculer la date d'expiration dynamiquement

Connexions persistantes

- Idée : autoriser le navigateur à faire plusieurs requêtes avec une même connexion TCP
  - Déjà fonctionnel
- Ne pas casser cette fonctionnalité existante
  - Fournir l'entête HTTP Content-Length par PHP
- Limiter le temps d'attente du serveur Web
  - Apache : KeepAliveTimeout 10
  - Éventuellement supprimer les keepalive si on ne sert que du PHP et qu'on le gère mal

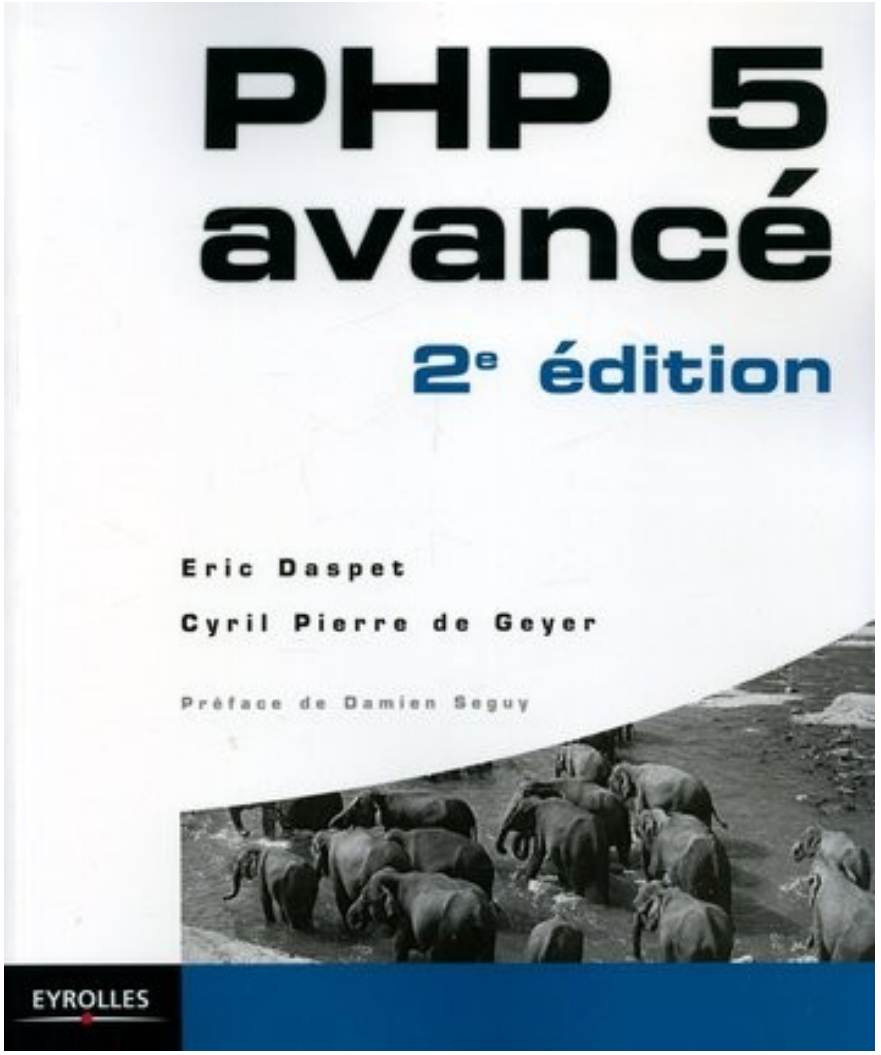
Plan de la session

- Avant-propos
- Configuration
- Code PHP
- Protocole HTTP
- Récapitulatif

Récapitulatif

- Manque de réseau ?
  - Compression, entêtes d'expiration, proxy
- Manque processeur ?
  - caches, 304, opcode, serveur Web, extensions C
- Manque de disque ? Trop d'IO ?
  - Tampons de sortie, configurations PHP et Apache
- Globalement, les réponses à tout faire :
  - Une gestion correcte du protocole HTTP ou acheter du matériel supplémentaire

PHP et performances



- Questions ?
- Expériences ?
- Partagez !

aston
architectes de la relation internet

sqli

- Demandez des infos

Eric.Daspet (à) survol.net