

Etude des principales solutions d'optimisation d'un serveur Web Apache / PHP / MySQL

Mise en œuvre
du Zend Optimizer, Zend Cache,
APC
et
jpcache



Armel FAUVEAU
armel.fauveau@globalis-ms.com

GLOBALIS media systems
<http://www.globalis-ms.com>

29 août 2001 – Version 1.1

1. INTRODUCTION.....	5
2. PRÉSENTATION DES SOLUTIONS D'OPTIMISATION.....	5
2.1 LES OPTIMISEURS	5
2.1.1 Principe	5
2.1.2 Solution testée : Zend Optimizer 1.1.0 (glibc2.1).....	5
2.2 LES CACHE D'OPCODE	6
2.2.1 Principe	6
2.2.2 Solutions testées : Zend Cache 1.1.0b (glibc2.1) et APC 1.1.0pl1	6
2.2.3 Solutions non testées	7
2.3 LES CACHES DE PAGES	7
2.3.1 Principe	7
2.3.2 Solution testée : jpcache 1.1.1	7
2.3.3 Solutions non testées	7
3. PRÉSENTATION DE LA PLATE-FORME DE TEST.....	8
3.1 SERVEUR.....	8
3.1.1 Côté hardware	8
3.1.2 Côté software	8
3.2 CLIENT	8
3.2.1 Côté hardware	8
3.2.2 Côté software	8
3.3 RÉSEAU	9
3.4 OUTIL DE MESURE.....	9
4. PRÉSENTATION DES SCRIPTS	9
4.1 SCRIPT A : HOMEPAGE DE PHPINDEX	9
4.2 SCRIPT B : RECHERCHE DANS L'ANNUAIRE DE PHPINDEX	9
4.3 SCRIPT C : CALCUL DES DÉCIMALES DE PI.....	10
5. PRÉSENTATION DU PROTOCOLE DE TEST.....	10
6. LECTURE DES TABLEAUX DE MESURES	10
7. PHP3.....	11
7.1 RÉSULTATS DES MESURES	11
7.1.1 Script A : homepage de PHPIndex.....	11
7.1.2 Script B : recherche dans l'annuaire de PHPIndex.....	11
7.1.3 Script C : calcul des décimales de PI.....	11
7.2 CONFIGURATION.....	12
7.3 ANALYSE	12
8. PHP4.....	13
8.1 RÉSULTATS DES MESURES	13
8.1.1 Script A : homepage de PHPIndex.....	13
8.1.2 Script B : recherche dans l'annuaire de PHPIndex.....	13
8.1.3 Script C : calcul des décimales de PI.....	13
8.2 CONFIGURATION.....	14
8.3 ANALYSE	14
9. PHP4+ZENDOPTIMIZER.....	16
9.1 RÉSULTATS DES MESURES	16

9.1.1	Script A : homepage de PHPIndex.....	16
9.1.2	Script B : recherche dans l'annuaire de PHPIndex.....	16
9.1.3	Script C : calcul des décimales de PI.....	16
9.2	CONFIGURATION.....	17
9.3	ANALYSE.....	17
10.	PHP4+ZENDCACHE.....	19
10.1	RÉSULTATS DES MESURES.....	19
10.1.1	Script A : homepage de PHPIndex.....	19
10.1.2	Script B : recherche dans l'annuaire de PHPIndex.....	19
10.1.3	Script C : calcul des décimales de PI.....	19
10.2	CONFIGURATION.....	20
10.3	ANALYSE.....	20
11.	PHP4+ZENDCACHE+ZENDOPTIMIZER.....	22
11.1	RÉSULTATS DES MESURES.....	22
11.1.1	Script A : homepage de PHPIndex.....	22
11.1.2	Script B : recherche dans l'annuaire de PHPIndex.....	22
11.1.3	Script C : calcul des décimales de PI.....	22
11.2	CONFIGURATION.....	23
11.3	ANALYSE.....	23
12.	PHP4+ALTERNATIVE PHP CACHE (MODE SHM).....	25
12.1	RÉSULTATS DES MESURES.....	25
12.1.1	Script A : homepage de PHPIndex.....	25
12.1.2	Script B : recherche dans l'annuaire de PHPIndex.....	25
12.1.3	Script C : calcul des décimales de PI.....	25
12.2	CONFIGURATION.....	26
12.3	ANALYSE.....	26
13.	PHP4+ALTERNATIVE PHP CACHE (MODE MMAP).....	28
13.1	RÉSULTATS DES MESURES.....	28
13.1.1	Script A : homepage de PHPIndex.....	28
13.1.2	Script B : recherche dans l'annuaire de PHPIndex.....	28
13.1.3	Script C : calcul des décimales de PI.....	28
13.2	CONFIGURATION.....	29
13.3	ANALYSE.....	29
14.	PHP4+JPCACHE (FICHER / TIMEOUT 900).....	31
14.1	RÉSULTATS DES MESURES.....	31
14.1.1	Script A : homepage de PHPIndex.....	31
14.1.2	Script B : recherche dans l'annuaire de PHPIndex.....	31
14.1.3	Script C : calcul des décimales de PI.....	31
14.2	CONFIGURATION.....	32
14.3	ANALYSE.....	32
PHP4+JPCACHE (FICHER / TIMEOUT 10).....		34
14.4	RÉSULTATS DES MESURES.....	34
14.4.1	Script A : homepage de PHPIndex.....	34
14.4.2	Script B : recherche dans l'annuaire de PHPIndex.....	34
14.4.3	Script C : calcul des décimales de PI.....	34
14.5	CONFIGURATION.....	35
14.6	ANALYSE.....	35

15. PHP4+JPCACHE (BASE / TIMEOUT 900)	37
15.1 RÉSULTATS DES MESURES	37
15.1.1 Script A : homepage de PHPIndex	37
15.1.2 Script B : recherche dans l'annuaire de PHPIndex	37
15.1.3 Script C : calcul des décimales de Pi	37
15.2 CONFIGURATION	38
15.3 ANALYSE	38
16. PHP4+JPCACHE (BASE / TIMEOUT 10)	40
16.1 RÉSULTATS DES MESURES	40
16.1.1 Script A : homepage de PHPIndex	40
16.1.2 Script B : recherche dans l'annuaire de PHPIndex	40
16.1.3 Script C : calcul des décimales de Pi	40
16.2 CONFIGURATION	41
16.3 ANALYSE	41
17. PHP4+ZENDCACHE+ZENDOPTIMIZER+JPCACHE (FICHER / TIMEOUT 10)	43
17.1 RÉSULTATS DES MESURES	43
17.1.1 Script A : homepage de PHPIndex	43
17.1.2 Script B : recherche dans l'annuaire de PHPIndex	43
17.1.3 Script C : calcul des décimales de Pi	43
17.2 CONFIGURATION	44
17.3 ANALYSE	44
18. CONCLUSION	46
A VERS LA STANDARDISATION D'UN BENCHMARK WEB	47
B A PROPOS DE GLOBALIS MEDIA SYSTEMS	48
C RESSOURCES	49
D LE SCRIPT DE CALCUL DES DÉCIMALES DE PI	50

1. Introduction

Devant un nombre croissant de connexions, les performances des serveurs Web s'effondrent. Afin d'améliorer les temps de réponse, le premier réflexe est généralement de dissocier le serveur HTTP et la base de données. Une solution tout aussi courante consiste à augmenter le nombre de serveurs HTTP en utilisant des solutions de « load balancing ».

Si ces solutions se montrent généralement efficaces, elles restent néanmoins coûteuses en terme de matériel et d'hébergement. De plus, elles sont parfois complexes à mettre en œuvre.

Pourtant d'autres solutions trop souvent méconnues existent. C'est en particulier le cas des « optimiseurs » et des systèmes de cache (« opcode » ou fichiers).

GLOBALIS se propose de les étudier ici dans le cadre d'une étude portant sur l'utilisation d'un serveur Web de type Apache, PHP et MySQL.

2. Présentation des solutions d'optimisation

Ces solutions peuvent être regroupées en 3 catégories :

- les optimiseurs,
- les caches d'opcode,
- les caches de page.

2.1 Les optimiseurs

2.1.1 Principe

Ces outils tentent d'accélérer l'exécution des scripts en optimisant certaines séquences d'instructions. Par exemple, ils remplacent les post-incrémentations par des pré-incrémentations plus rapides, à chaque fois que cela est possible.

2.1.2 Solution testée : Zend Optimizer 1.1.0 (glibc2.1)

Il s'agit du seul outil de ce type disponible actuellement. Il est développé par Zend, concepteur du moteur de PHP4. Il est disponible pour de nombreuses architectures (Linux, Solaris, FreeBSD, Windows NT/2000) et fonctionne avec Apache ou IIS.

Zend Optimizer est gratuit et librement téléchargeable depuis le site de Zend :
<http://www.zend.com/store/products/zend-optimizer.php>

L'installation est simple, mais nécessite de pouvoir éditer le fichier php.ini et de redémarrer le serveur http.

Une documentation claire ainsi qu'une FAQ sont disponibles sur le site de Zend.

Précisons pour finir que le Zend Optimizer est sous licence Zend Optimizer.

2.2 Les cache d'opcode

2.2.1 Principe

Lors de l'exécution d'un script, PHP4 procède ainsi :

- il charge l'intégralité du script,
- il convertit les instructions en opcodes (sorte de langage intermédiaire entre le script et un exécutable),
- il exécute les instructions.

Lorsqu'un même script est appelé plusieurs fois, ces opérations sont renouvelées. Ceci est une perte de temps évidente.

Les solutions de cache d'opcode se proposent donc d'effectuer une mise en cache des opcodes, afin d'en garder une trace. Ceci permet à PHP4 de les exécuter directement.

2.2.2 Solutions testées : Zend Cache 1.1.0b (glibc2.1) et APC 1.1.0p1

Nous avons testé 2 solutions.

▪ **Zend Cache**

Il est développé par Zend, concepteur du moteur de PHP4. Il est disponible pour de nombreuses architectures (Linux, Solaris, FreeBSD) et ne fonctionne qu'avec Apache.

Zend Cache est payant. Il est néanmoins possible de l'évaluer librement pendant une période de 30 jours. Vous pouvez le télécharger depuis le site de Zend :

<http://www.zend.com/store/products/zend-cache.php>

L'installation est simple, mais nécessite une étape de compilation, de pouvoir éditer le fichier php.ini et de redémarrer le serveur http.

Une documentation claire ainsi qu'une FAQ sont disponibles sur le site de Zend.

Au 20 juin 2001, Zend Cache coûtait entre 1875 et 9000 US\$ selon les processeurs utilisés, mais le prix n'apparaît plus sur le site de l'éditeur. Ajoutons que Zend semble le proposer gratuitement en échange d'un bandeau de publicité placé sur le site accéléré.

Précisons pour finir que le Zend Cache est sous licence Zend Cache.

▪ **APC : Alternative PHP Cache**

Il est édité par la société Community Connect. Il est disponible sous Linux et FreeBSD et ne fonctionnent qu'avec Apache.

APC est librement téléchargeable depuis le site de Community Connect :

<http://apc.communityconnect.com/>

L'installation est simple, mais nécessite une étape de compilation, de pouvoir éditer le fichier php.ini et de redémarrer le serveur.

Une documentation claire ainsi qu'une FAQ sont disponibles sur le site Community Connect.

Précisons pour finir qu'APC est sous licence QPL (Q Public License) 1.0.

2.2.3 Solutions non testées

D'autres solutions comparables existent. On peut en particulier citer afterBURNER*Cache. Disponible sur <http://bwcache.bware.it/cache.htm>, il propose globalement les mêmes fonctionnalités que les deux solutions présentées précédemment. Mais il semble être moins performant d'après les quelques études déjà réalisées. De plus, la dernière version en ligne date de mai 2001 et semble ne pas supporter PHP 4.0.6.

2.3 Les caches de pages

2.3.1 Principe

Ces solutions se composent généralement d'un unique script PHP qu'il faut inclure dans les scripts que l'on désire mettre en cache. Techniquement, elles utilisent les capacités de « output buffering » introduites par PHP4. C'est à dire qu'elles récoltent les sorties générées par les scripts afin de les stocker dans un fichier ou dans une base de données.

Le principal inconvénient de ces solutions repose sur la nécessité d'éditer l'ensemble des scripts que l'on désire mettre en cache. Ainsi, à la différence des solutions de cache d'opcode, la mise en œuvre de cette solution n'est pas transparente pour le développeur. Notons aussi qu'elle est inadaptée à certains scripts effectuant des UPDATE ou des INSERT dans une base. Idem si le contenu d'une page est régulièrement modifié.

Cependant, cette solution est la seule qui ne nécessite pas l'édition du php.ini et/ou le redémarrage du serveur. Elle peut donc être intéressante dans le cas d'un hébergement mutualisé.

2.3.2 Solution testée : jpcache 1.1.1

Cette solution nécessite au minimum PHP 4.0.1 avec le support Zlib. Elle repose sur un simple script (en fait 2, suivant le mode de stockage : fichier ou base) à inclure dans chaque fichier à mettre en cache.

Le script jpcache est librement téléchargeable sur :
<http://www.weirdpier.com/jpcache/>

L'installation est simple.

La documentation est succincte mais suffisante.

Précisons pour finir que jpcache est sous licence GPL.

2.3.3 Solutions non testées

D'autres solutions comparables existent. On peut en particulier citer :

- phpCache : disponible sur <http://0x00.org/php/phpCache/>. Il comporte plusieurs idées intéressantes et semble également fonctionner avec PHP3 (qui ne supporte pourtant pas le « output buffering »).

- Toncarta : disponible sur <http://www.heyes-computing.net/misc/toncarta.cache.php>. Attention, Toncarta ne « cache » pas les headers (et donc les paramètres passés par la méthode POST).

3. Présentation de la plate-forme de test

3.1 Serveur

3.1.1 Côté hardware

Le serveur utilisé est un bi-PIII à 700 Mhz. Il dispose d'un disque dur IDE Western Digital (WD205AA) de 20 Go, de 256 Mo de mémoire (SDRAM PC 100) et d'une carte réseau RealTek RTL-8029.

3.1.2 Côté software

Le système d'exploitation utilisé est Linux (distribution Slackware 7.0 régulièrement mise à jour) avec un noyau 2.4.6 SMP. La version d'Apache utilisée est la 1.3.12 avec PHP3 3.0.18 et PHP4 4.0.6 en module dynamique (apxs) et versioning (les deux versions de PHP cohabitent en même temps). Enfin, la version utilisée de MySQL est la 3.23.39.

A titre informatif, voici les paramètres de compilation employés pour PHP3 et PHP4 :

PHP3 3.0.18	<code>./configure '--with-apxs=/usr/local/apache/bin/apxs' '--with-mysql' '--with-gd' '--with-ttf' '--with-imap' '--without-xml' '--with-ftp' '--with-zlib' '--enable-versioning' '--enable-track-vars'</code>
PHP4 4.0.6	<code>'./configure' '--with-apxs=/usr/local/apache/bin/apxs' '--with-mysql=/usr/local/' '--with-gd' '--enable-ftp' '--enable-sockets' '--with-freetype' '--with-xml' '--with-gettext' '--with-gzip' '--with-bz2' '--enable-versioning' '--enable-track-vars' '--with-png-dir=/usr/local/lib' '--with-zlib-dir=/usr/local/lib/' '--with-imap'</code>

Concernant Apache, quelques affinages ont été effectués au niveau du fichier httpd.conf. Ce fût en particulier le cas au niveau du nombre maximal de clients supportés (MaxClients). Voici quelques paramètres de configuration utilisés :

- MinSpareServers 5
- MaxSpareServers 10
- StartServers 5
- MaxClients 100

3.2 Client

3.2.1 Côté hardware

Le client utilisé est un 486-DX4 à 100 Mhz. Il dispose d'un disque dur IDE Quantum Fireball (WD540A) de 540 Mo, de 32 Mo de mémoire et d'une carte réseau RealTek RTL-8029. Notons que l'influence de la puissance de l'architecture cliente sur les résultats obtenues est négligeable. Il n'était donc pas utile de chercher à surdimensionner celle ci.

3.2.2 Côté software

Le client tourne sous FreeBSD 4.2-RELEASE pour des raisons arbitraires.

3.3 Réseau

Le réseau fonctionnait en 10 Mb. Si, à première vue, ceci pouvait s'avérer un facteur limitant, les tests ont montré que le réseau n'a jamais été saturé (contrôle vérifié au niveau du HUB équipé d'une jauge de montée en charge).

3.4 Outil de mesure

L'outil de mesure utilisé dans le cadre de ces tests est ApacheBench 1.3c (fourni avec Apache). Cet outil tourne au niveau du client.

4. Présentation des scripts

4.1 Script A : homepage de PHPIndex

Il semblait judicieux de faire figurer la page d'accueil du site PHPIndex (<http://www.phpindex.com>) dans les tests. Et ceci pour de multiples raisons :

- Cette page est souvent le point d'entrée logique des visiteurs : il semble donc intéressant de chercher à mesurer la charge engendrée par cette page et trouver des solutions afin de l'optimiser.
- Cette page est caractéristique d'un site dynamique à base PHP / MySQL : on y trouve un jeu d'include successifs permettant de construire progressivement la page (entête, colonne de gauche assurant la navigation, colonne centrale d'informations, colonne de droite contextuelle et pied de page). Les principales informations présentées sont lues dynamiquement dans la base. Il s'agit surtout de requêtes de type SELECT COUNT ou SELECT LIMIT.

La homepage de PHPIndex est visible sur :
<http://www.phpindex.com>

4.2 Script B : recherche dans l'annuaire de PHPIndex

Une partie du site PHPIndex utilise Ht://dig comme moteur de recherche. Mais l'annuaire dispose de son propre système de recherche. Il effectue tout simplement un SELECT LIKE dans la table annuaire afin de faire remonter les sites comportant l'occurrence cherchée dans le titre ou la description. Le résultat fait également apparaître la liste de catégories et des sous-catégories dans lesquelles des sites ont été trouvés. Ce script va donc solliciter fortement la base MySQL, une recherche de type SELECT LIKE étant fortement consommatrice de ressources.

La recherche portait sur l'occurrence « php » et le résultat est visible sur :
http://www.phpindex.com/annuaire/annuaire_recherche.php3?motclef=php

4.3 Script C : calcul des décimales de PI

L'idée était ici d'écrire un script effectuant du calcul intensif et ne faisant pas intervenir MySQL. Le calcul des décimales de PI semblait un bon exemple.

Le script est livré en fin de document. Le résultat est également visible sur :
<http://www.phpindex.com/pi.php3>

5. Présentation du protocole de test

Les 3 scripts ont été exécutés avec les combinaisons suivantes :

- PHP3
- PHP4
- PHP4 + ZendOptimizer
- PHP4 + ZendCache
- PHP4 + ZendOptimizer + ZendCache
- PHP4 + APC (mode shm)
- PHP4 + APC (mode mmap)
- PHP4 + jpcache (fichier / timeout 900)
- PHP4 + jpcache (fichier / timeout 10)
- PHP4 + jpcache (base / timeout 900)
- PHP4 + jpcache (base / timeout 10)
- PHP4 + ZendOptimizer + ZendCache + jpcache (fichier / timeout 10)

Pour chacune de ces combinaisons, des mesures ont été effectuées pour 250 requêtes et successivement 1, 5, 10, 15, 20, 25 et 50 accès concurrents. Exemple :

```
/usr/local/sbin/ab -H "Accept-Encoding: gzip, deflate" -n 250 -c 5  
http://ip/script.php
```

Ici, l'argument -H "Accept-Encoding: gzip, deflate" est employé afin d'émuler un client en mesure de recevoir un flux compressé (fonctionnalité implémentée dans la plupart des clients et judicieusement exploitée par jpcache).

Une pose de 15 secondes a été systématiquement observée entre chaque mesure.

Enfin, l'utilitaire Unix « top » fonctionnait sur le serveur afin de garder un œil sur la montée en charge CPU et mémoire.

6. Lecture des tableaux de mesures

La légende des abréviations employées dans les tableaux de mesures est la suivante :

CL	Concurrency Level	FR	Failed Requests
CR	Complete Requests	TT	Total Transferred (bytes)
T	Time taken for tests (seconds)	HT	HTML Transferred (bytes)
RS	Requests per Second	TR	Transfer Rate (kb/s received)

Précisons enfin que la ligne grisée dans les tableaux de mesures présente le meilleur résultat obtenu en terme de requêtes par seconde.

7. PHP3

7.1 Résultats des mesures

7.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	31.908	7.84	0	8731000	8686750	273.63
5	250	18.218	13.72	0	8817679	8772721	484.01
10	250	18.527	13.49	0	8886764	8840921	479.67
15	250	18.841	13.27	0	8968279	8921728	476.00
20	250	19.083	13.10	0	9027192	8979756	473.05
25	250	19.337	12.93	0	9116168	9068024	471.44
50	250	20.435	12.23	0	9619112	9566897	470.72

7.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	29.739	8.41	0	10907500	10863250	366.77
5	250	17.071	14.64	0	10951130	10906703	641.50
10	250	17.438	14.34	0	11096345	11050502	636.33
15	250	17.585	14.22	0	11179876	11133325	635.76
20	250	18.395	13.59	0	11345680	11299129	616.78
25	250	18.472	13.53	0	11488692	11440725	621.95
50	250	20.724	12.06	0	12420776	12369623	599.34

7.1.3 Script C : calcul des décimales de PI

CL	CR	T	RS	FR	TT	HT	TR
1	250	555.764	0.45	0	137000	92750	0.25
5	250	279.657	0.89	0	137000	92750	0.49
10	250	281.350	0.89	0	137000	92750	0.49
15	250	285.820	0.87	0	137000	92750	0.48
20	250	290.902	0.86	0	137000	92750	0.47
25	250	284.377	0.88	0	137000	92750	0.48
50	250	Time Out	Time Out	Time Out	Time Out	Time Out	Time Out

7.2 Configuration

PHP3.0.18

```
./configure '--with-apxs=/usr/local/apache/bin/apxs' '--with-mysql' '--with-gd' '--with-ttf' '--with-imap'  
'--without-xml' '--with-ftp' '--with-zlib' '--enable-versioning' '--enable-track-vars'
```

7.3 Analyse

Ces résultats obtenus serviront de valeurs de référence pour la suite des mesures.

Nous verrons rapidement qu'ils correspondent aux performances les plus basses. Notons également que le serveur n'a pas supporté la charge lors de la tentative de mesure de 250 requêtes jouées par 50 accès concurrents pour le script de calcul de PI. En particulier, la charge CPU était trop importante.

8. PHP4

8.1 Résultats des mesures

8.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	23.430	10.67	0	8825500	8781500	376.68
5	250	14.183	17.63	0	8860431	8815903	624.72
10	250	16.650	15.02	0	9021622	8976390	541.84
15	250	15.491	16.14	0	9088843	9042555	586.72
20	250	15.732	15.89	0	9176038	9128870	583.27
25	250	15.769	15.85	0	9257493	9209445	587.07
50	250	16.930	14.77	0	9725632	9673360	574.46

8.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	24.354	10.27	0	11020500	10976500	452.51
5	250	14.024	17.83	0	11023420	10979244	786.04
10	250	14.290	17.49	0	11294661	11249077	790.39
15	250	15.080	16.58	0	11330141	11284733	751.34
20	250	16.485	15.17	0	11414180	11368596	692.40
25	250	15.438	16.19	0	11023420	10979244	714.04
50	250	17.489	14.29	0	11854128	11806432	677.80

8.1.3 Script C : calcul des décimales de PI

CL	CR	T	RS	FR	TT	HT	TR
1	250	169.982	1.47	0	136750	92750	0.80
5	250	93.959	2.66	0	136750	92750	1.46
10	250	95.444	2.62	0	136750	92750	1.43
15	250	95.522	2.62	0	136750	92750	1.43
20	250	98.124	2.55	0	136750	92750	1.39
25	250	98.264	2.54	0	136750	92750	1.39
50	250	102.493	2.44	0	136750	92750	1.33

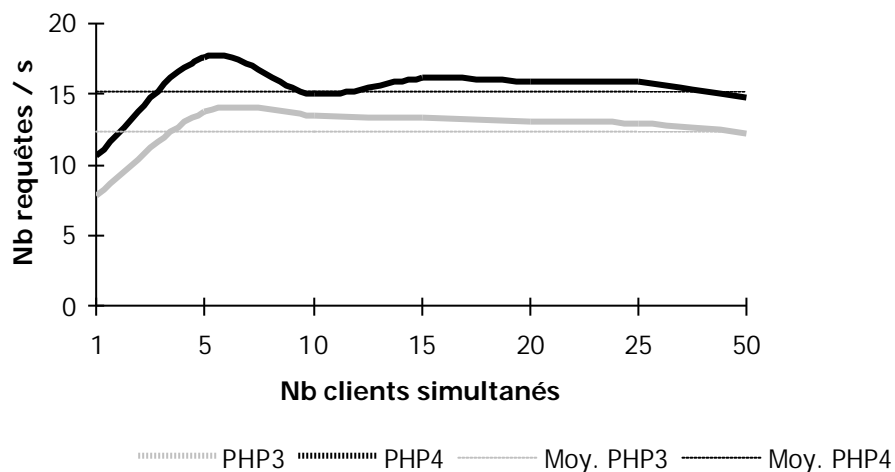
8.2 Configuration

PHP4.0.6

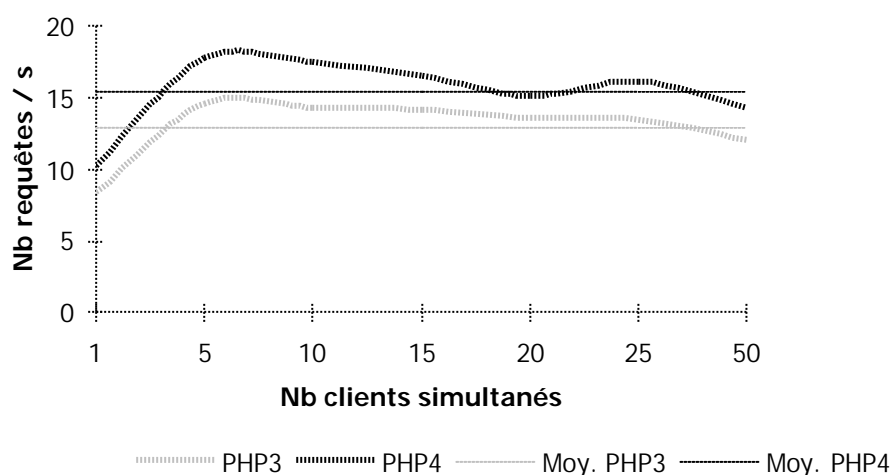
```
./configure' '--with-apxs=/usr/local/apache/bin/apxs' '--with-mysql=/usr/local/' '--with-gd' '--enable-ftp' '--enable-sockets' '--with-freetype' '--with-xml' '--with-gettext' '--with-gzip' '--with-bz2' '--enable-versioning' '--enable-track-vars' '--with-png-dir=/usr/local/lib' '--with-zlib-dir=/usr/local/lib/' '--with-imap'
```

8.3 Analyse

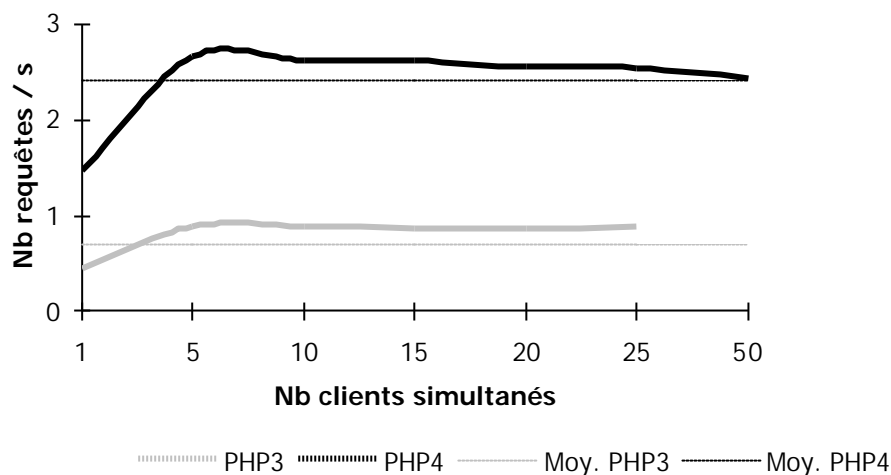
homepage de PHPIindex



Recherche dans l'annuaire de PHPIindex



Calcul des décimales de PI



Comme précédemment, ces résultats obtenus serviront de valeurs de référence pour la suite des mesures.

Il est déjà intéressant de noter l'amélioration sensible des performances dans tous les cas. Le gain est important, entre 20 et 200% suivant le type de script.

PHP4 résulte d'une réécriture complète de PHP3 et le « parser » Zend se montre très efficace.

9. PHP4+ZendOptimizer

9.1 Résultats des mesures

9.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	26.623	9.39	0	8825500	8781500	331.50
5	250	14.591	17.13	0	8932548	8887844	612.20
10	250	14.920	16.76	0	8988835	8943251	602.47
15	250	15.387	16.25	0	9080607	9034319	590.15
20	250	15.735	15.89	0	9130038	9083398	580.24
25	250	15.971	15.65	0	9311077	9262853	583.00
50	250	17.061	14.65	0	9780095	9727471	573.24

9.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	24.142	10.36	0	11020500	10976500	456.419
5	250	14.112	17.72	0	11149133	11104429	790.05
10	250	13.739	18.20	0	11161946	11116714	812.43
15	250	14.949	16.72	0	11111474	11066946	743.29
20	250	16.136	15.49	0	11417238	11371654	707.56
25	250	17.349	14.41	0	12018413	11970365	692.74
50	250	16.488	15.16	0	11387756	11341996	690.67

9.1.3 Script C : calcul des décimales de PI

CL	CR	T	RS	FR	TT	HT	TR
1	250	106.243	2.35	0	136750	92750	1.29
5	250	62.915	3.97	0	136750	92750	2.17
10	250	63.817	3.92	0	136750	92750	2.14
15	250	64.208	3.89	0	136750	92750	2.13
20	250	64.996	3.85	0	136750	92750	2.10
25	250	65.932	3.79	0	136750	92750	2.07
50	250	71.117	3.52	0	136750	92750	1.92

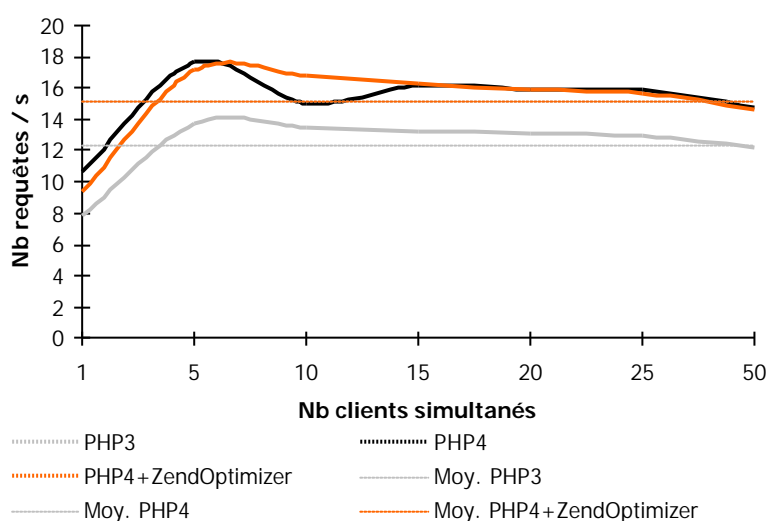
9.2 Configuration

PHP4.0.6 + ZendOptimizer-1.1.0 (Linux_glibc2.1)

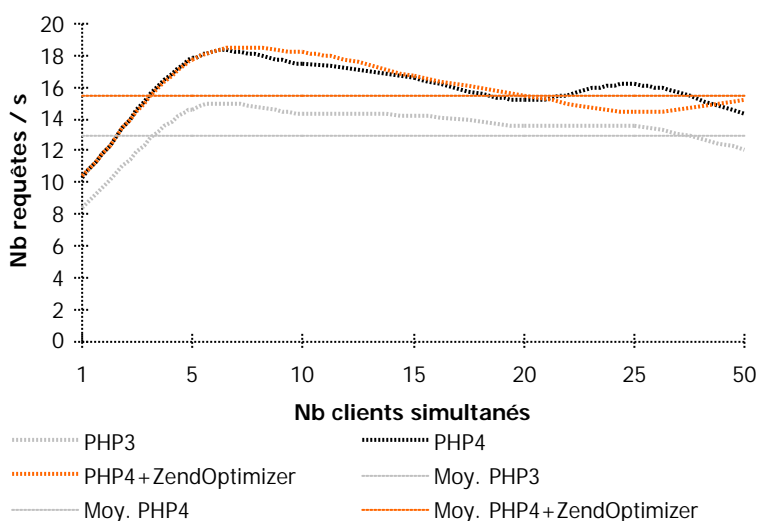
zend_optimizer.optimization_level=15
 zend_extension=/usr/local/Zend/lib/ZendOptimizer.so

9.3 Analyse

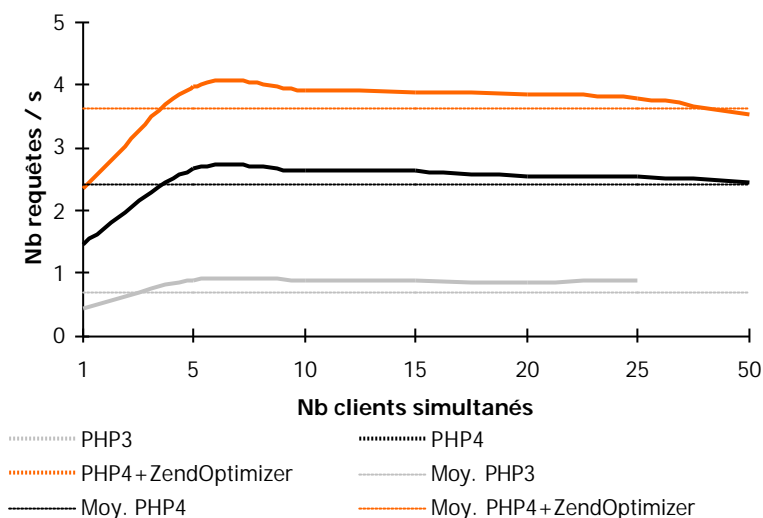
homepage de PHPIindex



Recherche dans l'annuaire de PHPIindex



Calcul des décimales de PI



Pas de réelle surprise ici.

L'utilisation du Zend Optimizer n'apporte rien pour les 2 premiers scripts. Les performances sont même très légèrement inférieures à celles de PHP4 seul. La tentative d'optimisation de ces scripts est donc ici pénalisante.

En revanche, l'impact est indéniable dans le cas du dernier script. Dans ce cas, le gain de performance est de 346% par rapport à PHP3 et de près de 50% par rapport à PHP4 seul.

10. PHP4+ZendCache

10.1 Résultats des mesures

10.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	22.033	11.35	0	8825500	8781500	400.56
5	250	12.086	20.69	0	8894290	8849586	735.92
10	250	12.370	20.21	0	8986769	8941185	726.50
15	250	12.926	19.34	0	9095359	9049071	703.65
20	250	13.149	19.01	0	9224648	9177304	701.55
25	250	13.213	18.92	0	9260779	9212907	700.88
50	250	14.087	17.75	0	9758532	9705908	692.73

10.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	20.265	12.34	0	11020500	10976500	543.82
5	250	13.291	18.81	0	11064582	11020406	832.49
10	250	13.517	18.50	0	11184584	11139528	827.45
15	250	13.917	17.96	0	11285324	11239388	810.90
20	250	14.258	17.53	0	11108664	11064312	779.12
25	250	15.526	16.10	0	11939978	11892282	769.03
50	250	16.861	14.83	0	12012132	11961268	712.42

10.1.3 Script C : calcul des décimales de PI

CL	CR	T	RS	FR	TT	HT	TR
1	250	170.015	1.47	0	136750	92750	0.80
5	250	93.960	2.66	0	136750	92750	1.46
10	250	94.917	2.63	0	136750	92750	1.44
15	250	96.559	2.59	0	136750	92750	1.42
20	250	97.759	2.56	0	136750	92750	1.40
25	250	98.605	2.54	0	136750	92750	1.39
50	250	103.348	2.42	0	136750	92750	1.32

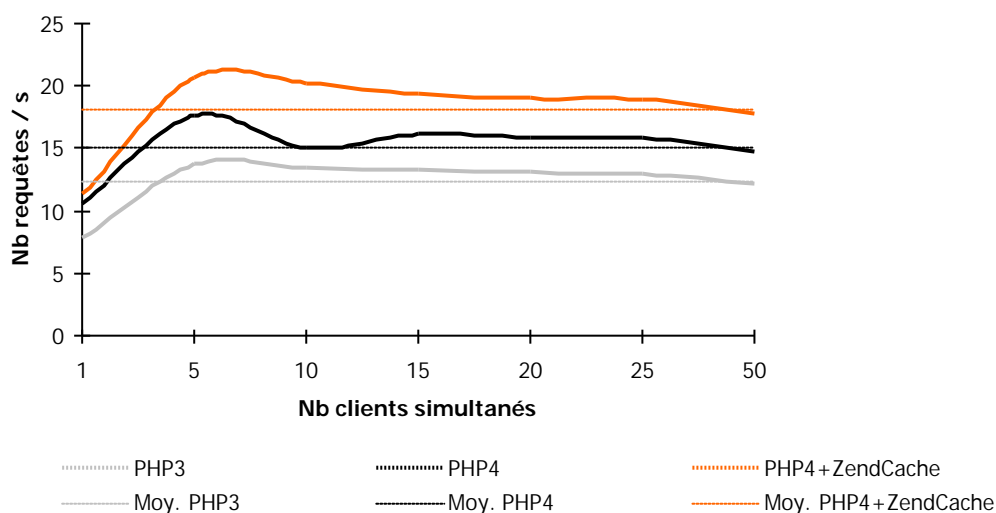
10.2 Configuration

PHP4.0.6 + ZendCache-1.1.0b (Linux_glibc2.1)

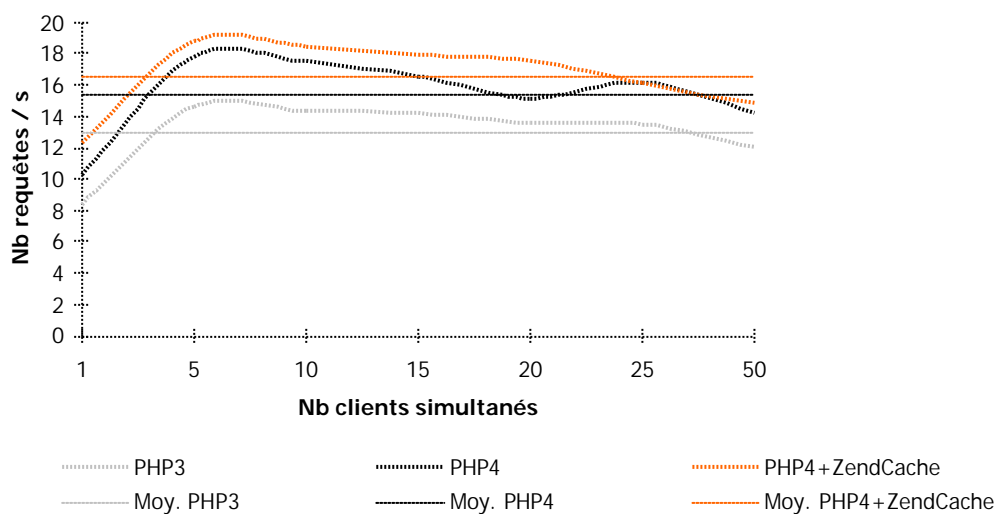
```
zend_cache.memory_consumption=64  
zend_cache.validate_timestamps=1  
zend_cache.use_cwd=1  
zend_extension=/usr/local/Zend/lib/ZendCache.so
```

10.3 Analyse

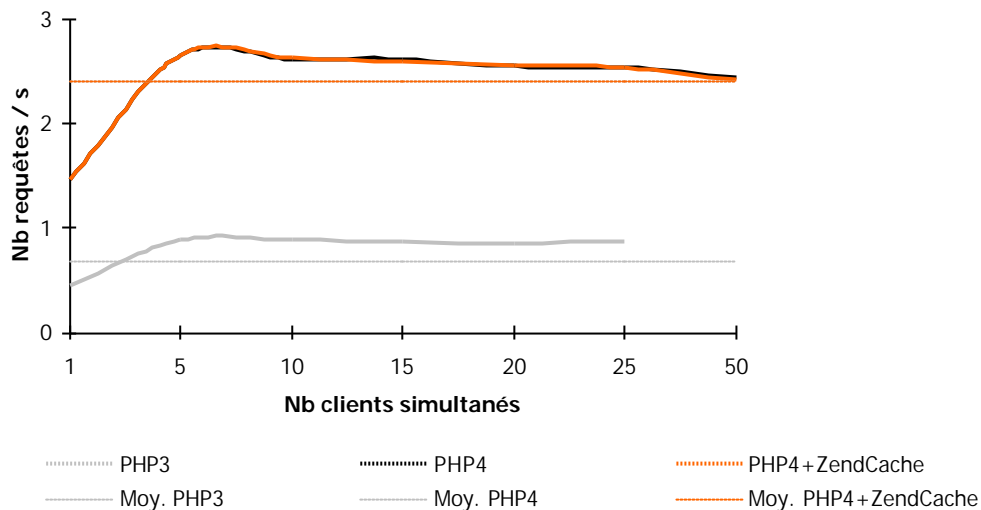
homepage de PHPIindex



Recherche dans l'annuaire de PHPIindex



Calcul des décimales de PI



La situation est inversée.

Le Zend Cache permet d'améliorer sensiblement les résultats sur les 2 premiers scripts. En revanche, il est sans effet sur le dernier.

11. PHP4+ZendCache+ZendOptimizer

11.1 Résultats des mesures

11.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	19.852	12.59	0	8825500	8781500	444.56
5	250	11.912	20.99	0	8901312	8856608	747.26
10	250	11.958	20.91	0	8992542	8947310	752.01
15	250	12.397	20.17	0	9086560	9040624	732.96
20	250	12.920	19.35	0	9198172	9150828	711.93
25	250	13.187	18.96	0	9289587	9241715	704.45
50	250	14.120	17.71	0	9747992	9695544	690.37

11.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	18.879	13.24	0	11020500	10976500	583.74
5	250	13.372	18.70	0	11080642	11036114	828.65
10	250	13.690	18.26	0	11243632	11198224	821.30
15	250	14.200	17.61	0	11375652	11330068	801.10
20	250	14.657	17.06	0	11373156	11327748	775.95
25	250	15.599	16.03	0	11939978	11892282	765.43
50	250	16.214	15.42	0	11827612	11778332	729.47

11.1.3 Script C : calcul des décimales de PI

CL	CR	T	RS	FR	TT	HT	TR
1	250	105.299	2.37	0	136750	92750	1.30
5	250	62.747	3.98	0	136750	92750	2.18
10	250	63.234	3.95	0	136750	92750	2.16
15	250	63.864	3.91	0	136750	92750	2.14
20	250	65.027	3.84	0	136750	92750	2.10
25	250	65.369	3.82	0	136750	92750	2.09
50	250	67.264	3.72	0	136750	92750	2.03

11.2 Configuration

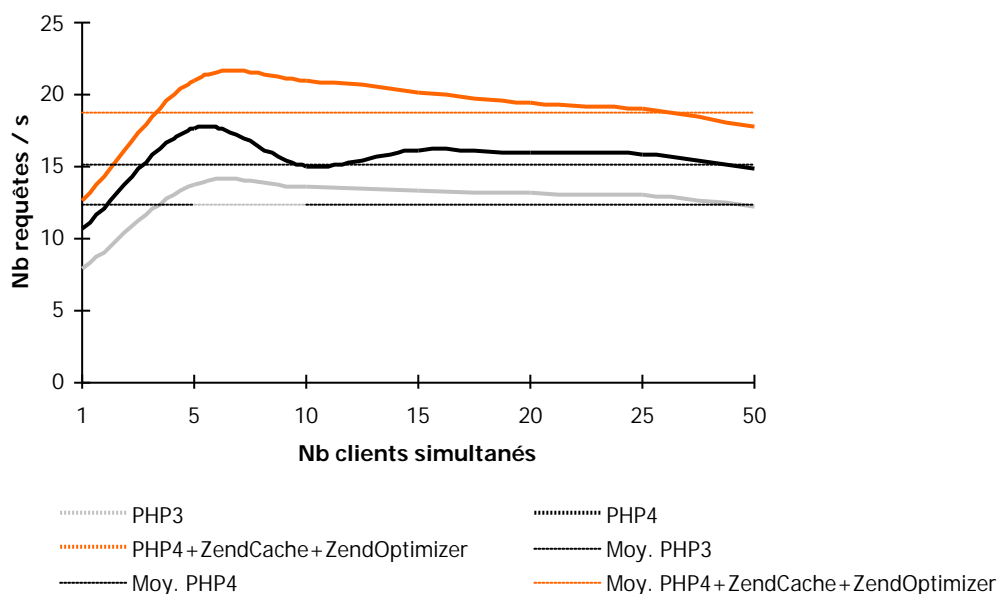
PHP4.0.6 + ZendOptimizer-1.1.0 + ZendCache-1.1.0b (Linux_glibc2.1)

```
zend_optimizer.optimization_level=15  
zend_extension=/usr/local/Zend/lib/ZendOptimizer.so
```

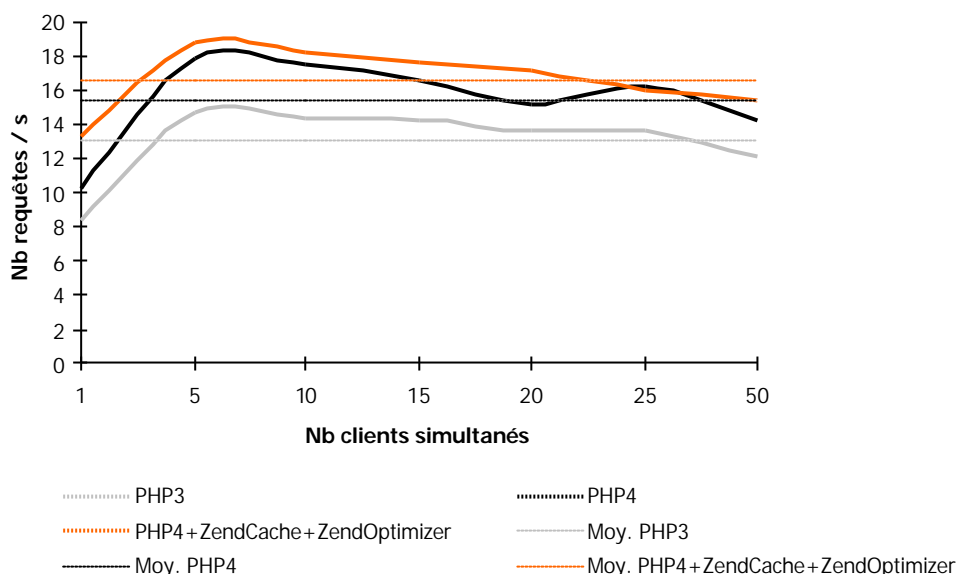
```
zend_cache.memory_consumption=64  
zend_cache.validate_timestamps=1  
zend_cache.use_cwd=1  
zend_extension=/usr/local/Zend/lib/ZendCache.so
```

11.3 Analyse

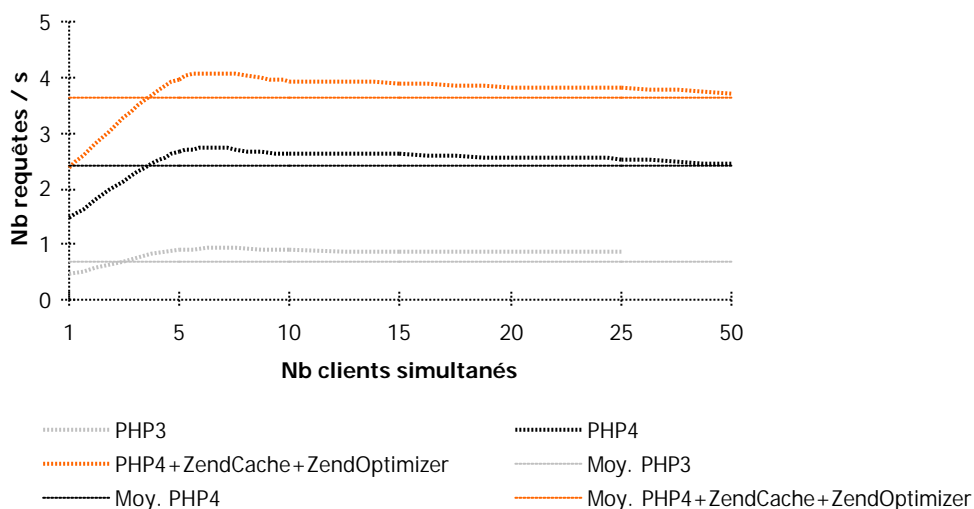
homepage de PHPIindex



Recherche dans l'annuaire de PHPIndex



Calcul des décimales de PI



Il est parfaitement possible de combiner le Zend Cache et le Zend Optimizer.

Cette combinaison s'avère intéressante puisqu'elle permet d'obtenir de bons résultats pour l'ensemble des scripts. Les performances sont nettement améliorées.

Zend Cache et Zend Optimizer sont complémentaires.

12. PHP4+Alternative PHP Cache (mode SHM)

12.1 Résultats des mesures

12.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	21.937	11.40	0	8825500	8781500	402.31
5	250	13.407	18.65	0	8887209	8842681	662.88
10	250	14.209	17.59	0	8959119	8913535	630.52
15	250	14.768	16.93	0	9064113	9017825	613.77
20	250	15.173	16.48	0	9163942	9116598	603.96
25	250	15.432	16.20	0	9262411	9214187	600.21
50	250	16.523	15.13	0	9767763	9715315	591.16

12.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	22.670	11.03	0	11020500	10976500	486.13
5	250	13.817	18.09	0	11086230	11041526	802.36
10	250	14.165	17.65	0	11364859	11319275	802.32
15	250	14.947	16.73	0	11530862	11484750	771.45
20	250	15.298	16.34	0	11464240	11418304	749.39
25	250	15.995	15.63	0	11920838	11873142	745.29
50	250	16.316	15.32	0	11980862	11932286	734.30

12.1.3 Script C : calcul des décimales de PI

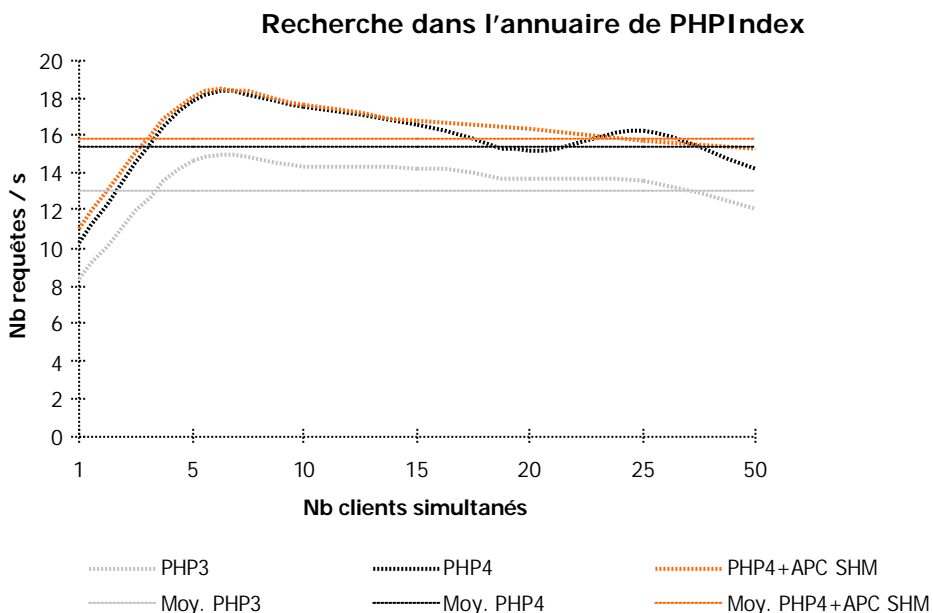
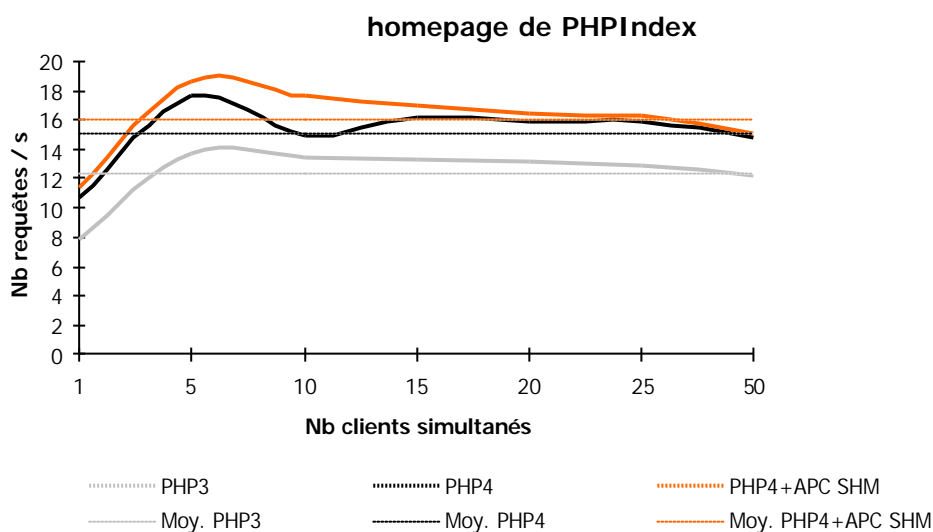
CL	CR	T	RS	FR	TT	HT	TR
1	250	169.986	1.47	0	136750	92750	0.80
5	250	94.764	2.64	0	136750	92750	1.44
10	250	96.099	2.60	0	136750	92750	1.42
15	250	96.907	2.58	0	136750	92750	1.41
20	250	97.401	2.57	0	136750	92750	1.40
25	250	99.080	2.52	0	136750	92750	1.38
50	250	103.577	2.41	0	136750	92750	1.32

12.2 Configuration

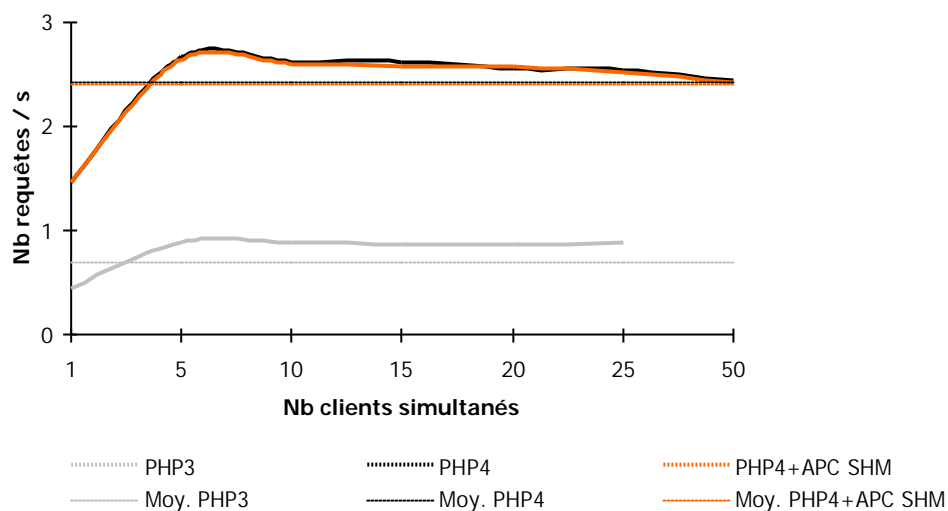
PHP4.0.6 + APC 1.1.0pl1

zend_extension=/usr/local/Zend/lib/php_apc.so
 apc.mode = shm

12.3 Analyse



Calcul des décimales de PI



APC est efficace sur les 2 premiers scripts et sans effet sur le dernier script. Les résultats sont donc semblables à ceux du Zend cache tout en restant, dans tous les cas, inférieurs. Cette différence atteint 10% dans le cas de la homepage.

13. PHP4+Alternative PHP Cache (mode MMAP)

13.1 Résultats des mesures

13.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	23.540	10.62	0	8825500	8781500	374.92
5	250	13.746	18.19	0	8926901	8882197	649.42
10	250	14.086	17.75	0	8984793	8939209	637.85
15	250	14.615	17.11	0	9138897	9092433	625.31
20	250	14.807	16.88	0	9171266	9123922	619.39
25	250	14.500	17.24	0	8951846	8907142	617.37
50	250	15.998	15.63	0	9619468	9567724	601.29

13.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	22.384	11.17	0	11020500	10976500	492.34
5	250	13.763	18.16	0	11093885	11049181	806.07
10	250	14.352	17.42	0	11340144	11294560	790.14
15	250	15.326	16.31	0	11600792	11554328	756.94
20	250	15.290	16.35	0	11365608	11320200	743.34
25	250	16.453	15.19	0	12025326	11977278	730.89
50	250	16.239	15.40	0	12011418	11963370	739.66

13.1.3 Script C : calcul des décimales de PI

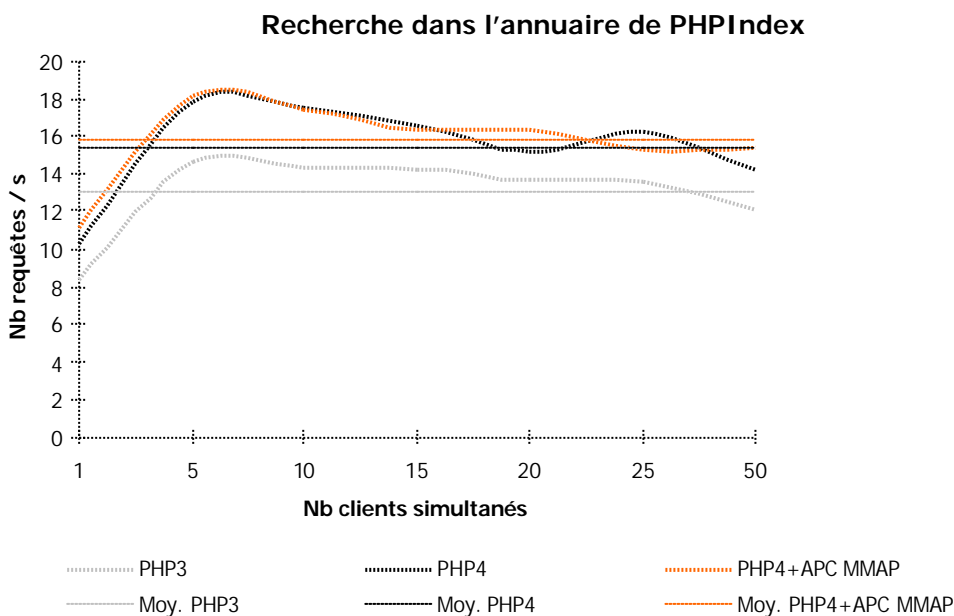
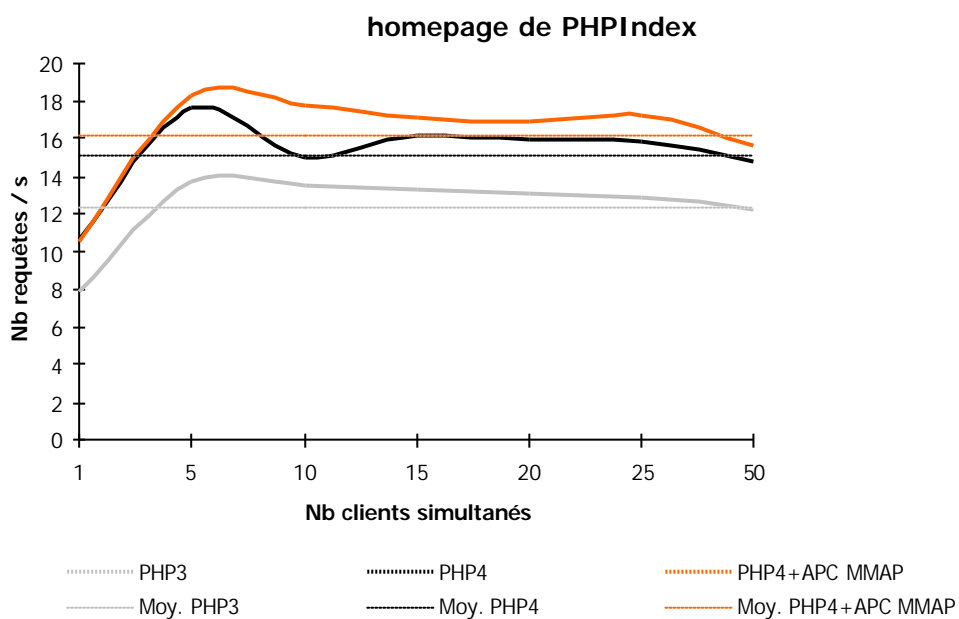
CL	CR	T	RS	FR	TT	HT	TR
1	250	169.996	1.47	0	136750	92750	0.80
5	250	94.559	2.64	0	136750	92750	1.45
10	250	96.002	2.60	0	136750	92750	1.42
15	250	96.738	2.58	0	136750	92750	1.41
20	250	97.827	2.56	0	136750	92750	1.40
25	250	98.409	2.54	0	136750	92750	1.39
50	250	102.568	2.44	0	136750	92750	1.33

13.2 Configuration

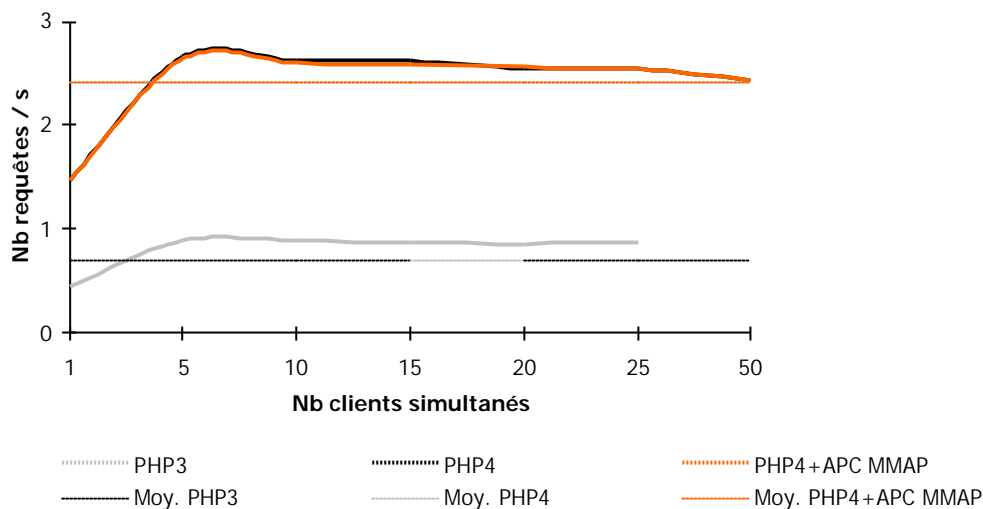
PHP4.0.6 + APC 1.1.0pl1

zend_extension=/usr/local/Zend/lib/php_apc.so
 apc.mode = mmap

13.3 Analyse



Calcul des décimales de PI



Idem que précédemment. Le mode mmap n'a pas permis d'obtenir des résultats meilleurs que ceux observés avec le Zend Cache.

Notons aussi que l'APC charge beaucoup plus le serveur, comparé au Zend Cache, en terme d'occupation CPU et mémoire.

14. PHP4+jpcache (fichier / timeout 900)

14.1 Résultats des mesures

14.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	7.515	33.27	0	1251500	1188250	166.53
5	250	2.886	86.63	0	1251500	1188250	433.65
10	250	2.771	90.22	0	1269438	1205176	458.12
15	250	2.803	89.19	0	1277364	1212596	455.71
20	250	2.812	88.90	0	1270272	1205757	451.73
25	250	2.928	85.38	0	1270272	1205757	433.84
50	250	2.945	84.89	0	1284038	1218511	436.01

14.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	6.821	36.65	0	1141750	1078000	167.39
5	250	2.654	94.20	0	1149237	1084977	433.02
10	250	2.627	95.17	0	1167131	1101596	444.28
15	250	2.553	97.92	0	1164211	1098931	456.02
20	250	2.630	95.06	0	1179185	1112885	448.36
25	250	2.607	95.90	0	1173345	1107555	450.07
50	250	2.789	89.64	0	1173345	1107555	420.70

14.1.3 Script C : calcul des décimales de PI

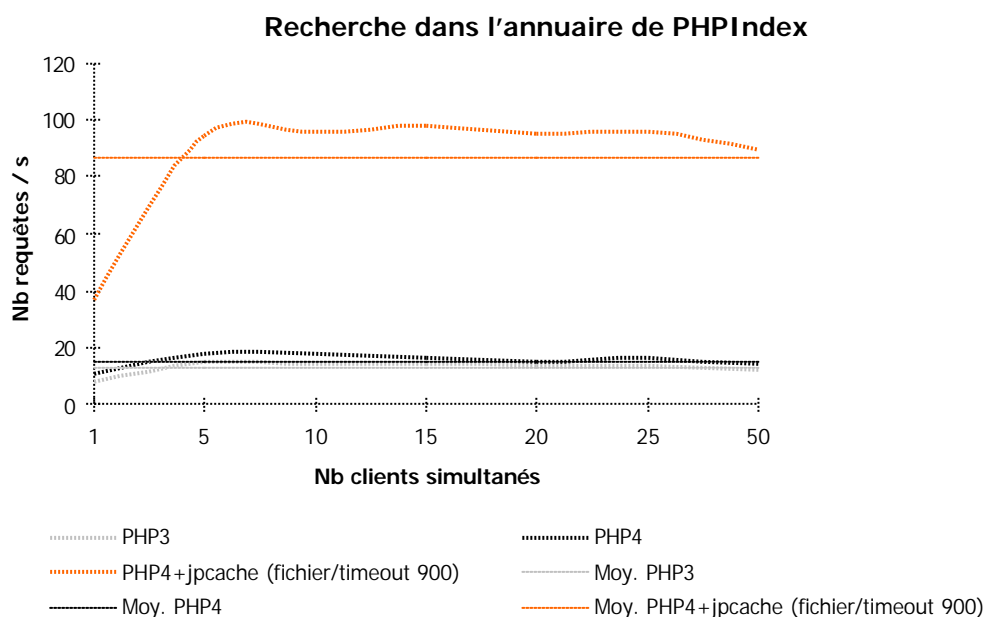
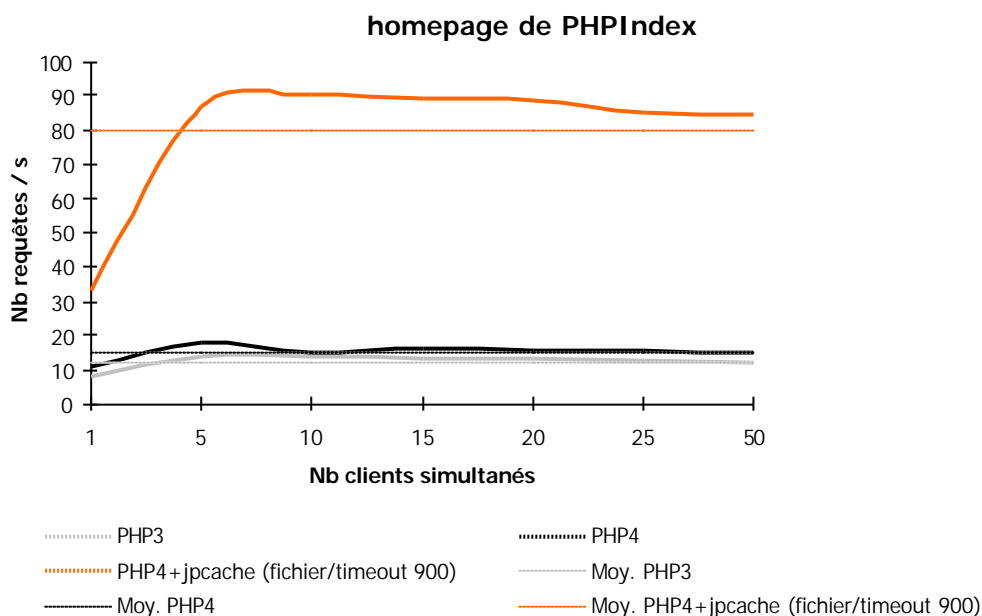
CL	CR	T	RS	FR	TT	HT	TR
1	250	4.001	62.48	0	112500	49000	28.12
5	250	1.624	153.94	0	112500	49000	69.27
10	250	1.613	154.99	0	112500	49000	69.75
15	250	1.566	159.64	0	112500	49000	71.84
20	250	1.632	153.19	0	112950	49196	69.21
25	250	1.634	153.00	0	113400	49392	69.40
50	250	1.678	148.99	0	113400	49392	67.58

14.2 Configuration

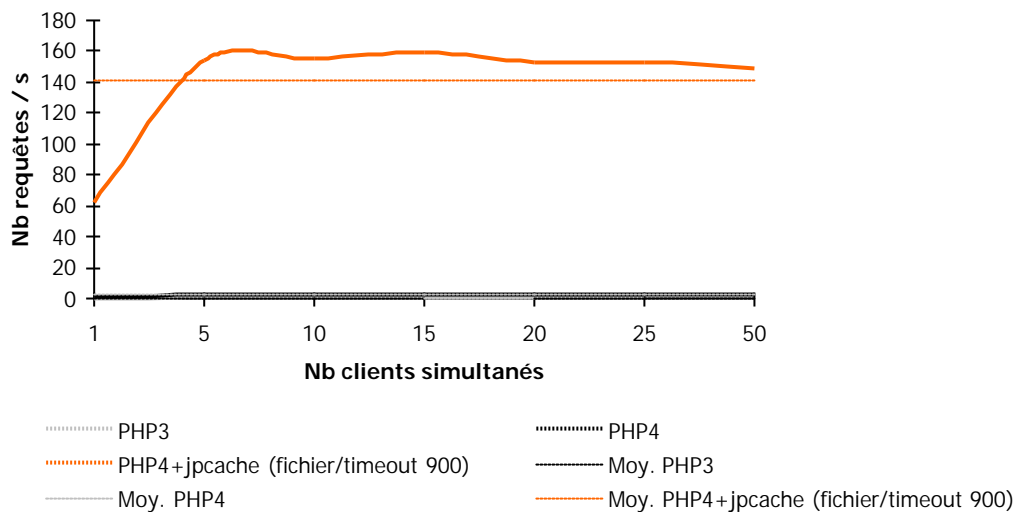
PHP4.0.6 + jpcache 1.1.1 (jpcache.inc)

\$cachetimeout=900

14.3 Analyse



Calcul des décimales de PI



Les résultats obtenus sont excellents et même spectaculaires pour l'ensemble des scripts. On note une amélioration des performances par rapport au PHP3 comprises entre 530 et 17200% !

Attention néanmoins. Une mise en cache de 900 secondes (soit 15 minutes) n'est pas vraiment réaliste dans le cas du test, une seule mise en cache étant effectuée au début pour l'ensemble des mesures (qui durent moins de 15 minutes). Il semblait donc intéressant de rejouer cette combinaison avec un time out plus fin (10 secondes).

Précisons enfin le point suivant : jpcache compresse le fichier mis en cache. Et c'est la version compressée qui est envoyée au client. Si le client ne supporte pas la compression, jpcache va se charger de la faire pour lui, opération coûteuse en terme de ressources CPU. Dans ce cas, les mesures observées (non présentées ici), sont proches de PHP4 seul.

PHP4+jpcache (fichier / timeout 10)

14.4 Résultats des mesures

14.4.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	7.513	33.28	0	1251500	1188250	166.58
5	250	2.867	87.20	0	1256506	1193003	438.27
10	250	2.804	89.16	0	1262346	1198337	450.19
15	250	2.809	89.00	0	1272358	1207843	452.96
20	250	2.806	89.09	0	1271524	1207262	453.14
25	250	2.784	89.80	0	1262346	1198337	453.43
50	250	2.920	85.62	0	1277364	1212596	437.45

14.4.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	6.936	36.04	0	1141750	1078000	164.61
5	250	2.771	90.22	0	1144670	1080665	413.09
10	250	2.666	93.77	0	1170425	1104890	439.02
15	250	2.700	92.59	0	1165858	1100578	431.80
20	250	2.743	91.14	0	1189966	1123156	433.82
25	250	2.857	87.50	0	1191613	1124803	417.09
50	250	2.836	88.15	0	1170425	1104890	412.70

14.4.3 Script C : calcul des décimales de PI

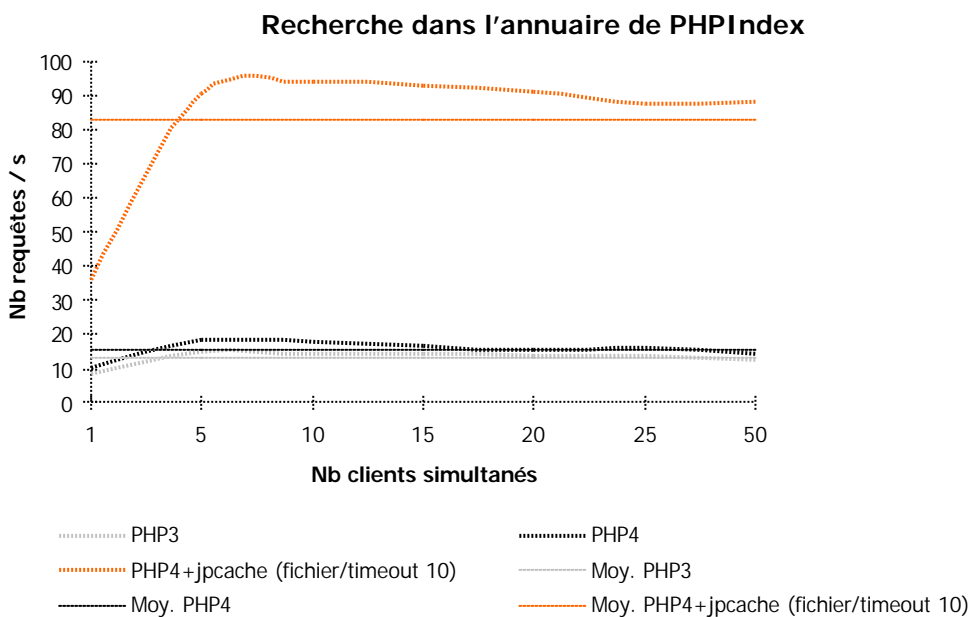
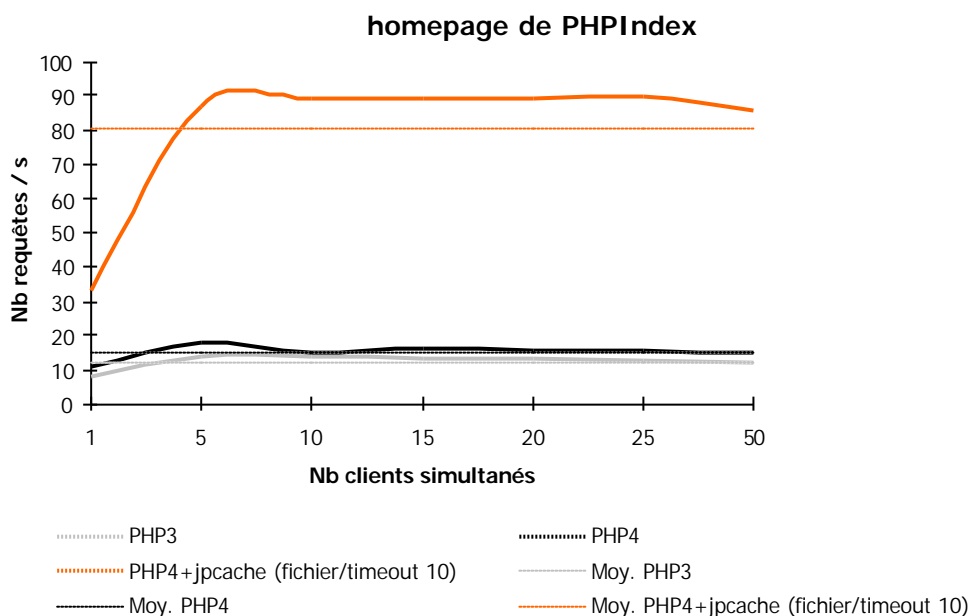
CL	CR	T	RS	FR	TT	HT	TR
1	250	4.683	53.38	0	112500	49000	24.02
5	250	2.378	105.13	0	112500	49000	47.31
10	250	3.012	83.00	0	112950	49196	37.50
15	250	3.039	82.26	0	115650	50372	38.06
20	250	2.328	107.39	0	113850	49588	48.90
25	250	2.355	106.16	0	114750	49980	48.73
50	250	2.396	104.34	0	114300	49784	47.70

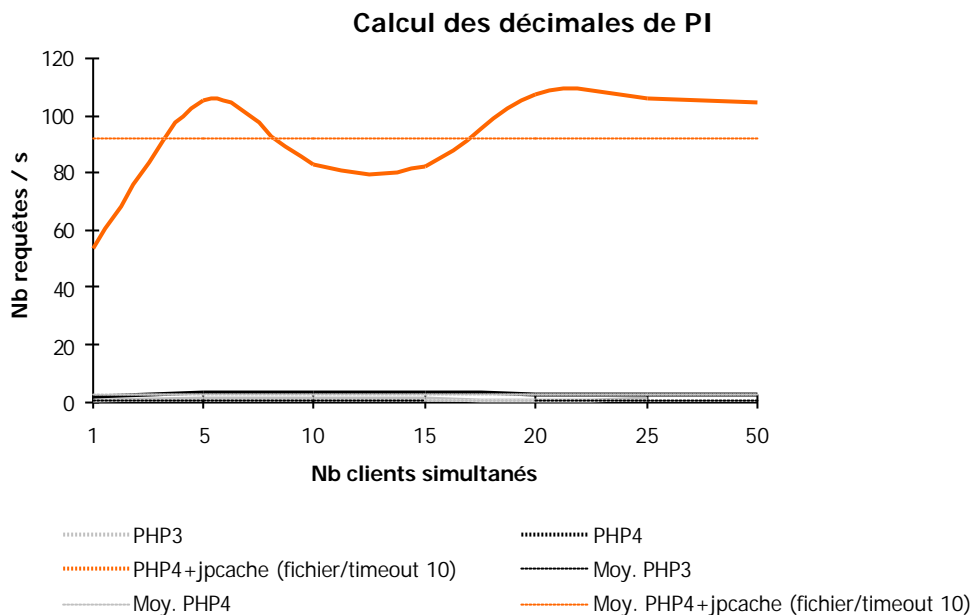
14.5 Configuration

PHP4.0.6 + jpcache 1.1.1 (jpcache.inc)

\$cachetimeout=10

14.6 Analyse





Les résultats obtenus restent excellents, malgré une légère baisse des performances. Le gain reste compris entre 510 et 11700% par rapport à PHP3.

15. PHP4+jpccache (base / timeout 900)

15.1 Résultats des mesures

15.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	7.642	32.71	0	1259000	1188250	164.75
5	250	4.057	61.62	0	1259000	1188250	310.33
10	250	4.060	61.58	0	1286592	1213578	316.89
15	250	3.832	65.24	0	1279144	1207262	333.81
20	250	3.813	65.57	0	1279144	1207262	335.47
25	250	3.989	62.67	0	1359720	1283310	340.87
50	250	3.909	63.95	0	1264036	1193003	323.37

15.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	6.871	36.38	0	1149250	1078000	167.26
5	250	3.833	65.22	0	1158444	1086624	302.23
10	250	3.642	68.64	0	1155090	1083270	317.16
15	250	3.558	70.26	0	1204414	1129744	338.51
20	250	3.552	70.38	0	1218205	1142680	342.96
25	250	3.545	70.52	0	1240756	1163521	350.00
50	250	3.366	74.27	0	1165961	1093571	346.39

15.1.3 Script C : calcul des décimales de PI

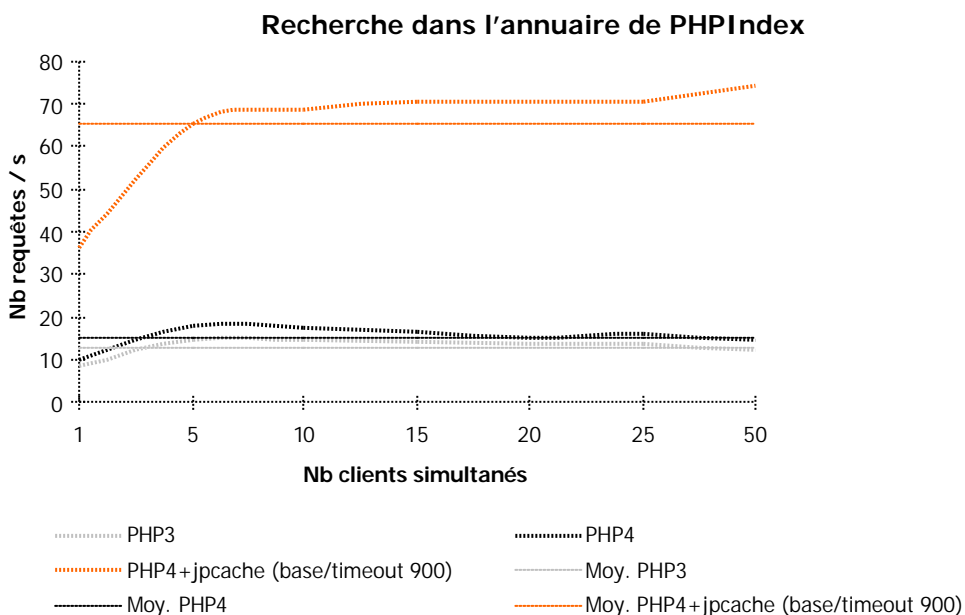
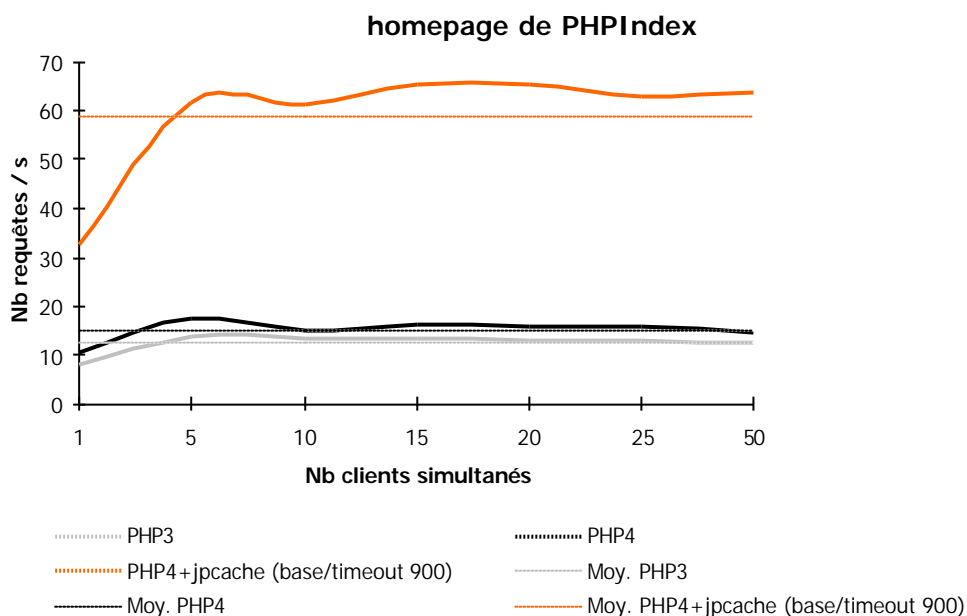
CL	CR	T	RS	FR	TT	HT	TR
1	250	4.618	54.14	0	120000	49000	25.99
5	250	2.186	114.36	0	121440	49588	55.55
10	250	2.095	119.33	0	123360	50372	58.88
15	250	2.044	122.31	0	120480	49196	58.94
20	250	2.100	119.05	0	120000	49000	57.14
25	250	2.034	122.91	0	124800	50960	61.36
50	250	2.204	113.43	0	120960	49392	54.88

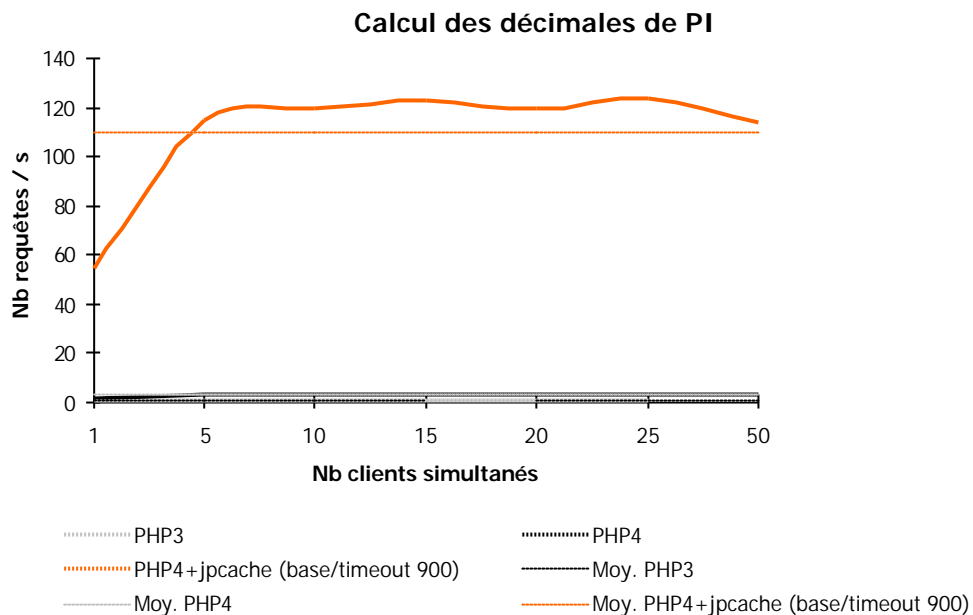
15.2 Configuration

PHP4.0.6 + jpcache 1.1.1 (jpcache-sql.php)

\$cachetimeout=900

15.3 Analyse





Les résultats obtenus sont bons ! Et ceci pour l'ensemble des scripts.

On note cependant que les performances sont beaucoup plus faibles lorsque jpgcache utilise une mise en cache au niveau d'une base de données (MySQL). Ce qui semble évident.

Là encore, il semblait intéressant de rejouer cette combinaison avec un time out plus fin (10 secondes).

16. PHP4+jpccache (base / timeout 10)

16.1 Résultats des mesures

16.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	7.596	32.91	0	1259000	1188250	165.75
5	250	4.206	59.44	0	1264036	1193003	300.53
10	250	4.376	57.13	0	1264036	1193003	288.86
15	250	4.455	56.12	0	1289216	1216768	289.39
20	250	4.980	50.20	0	1334540	1259545	267.98
25	250	5.117	48.86	0	1264840	1193524	247.18
50	250	5.185	48.22	0	1319432	1245286	254.47

16.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	6.832	36.59	0	1149250	1078000	168.22
5	250	3.851	64.92	0	1149250	1078000	298.43
10	250	4.095	61.05	0	1163041	1090936	284.01
15	250	4.428	56.46	0	1195220	1121120	269.92
20	250	4.133	60.49	0	1149250	1078000	278.07
25	250	4.841	51.64	0	1257901	1179811	259.84
50	250	5.126	48.77	0	1287160	1207360	251.10

16.1.3 Script C : calcul des décimales de PI

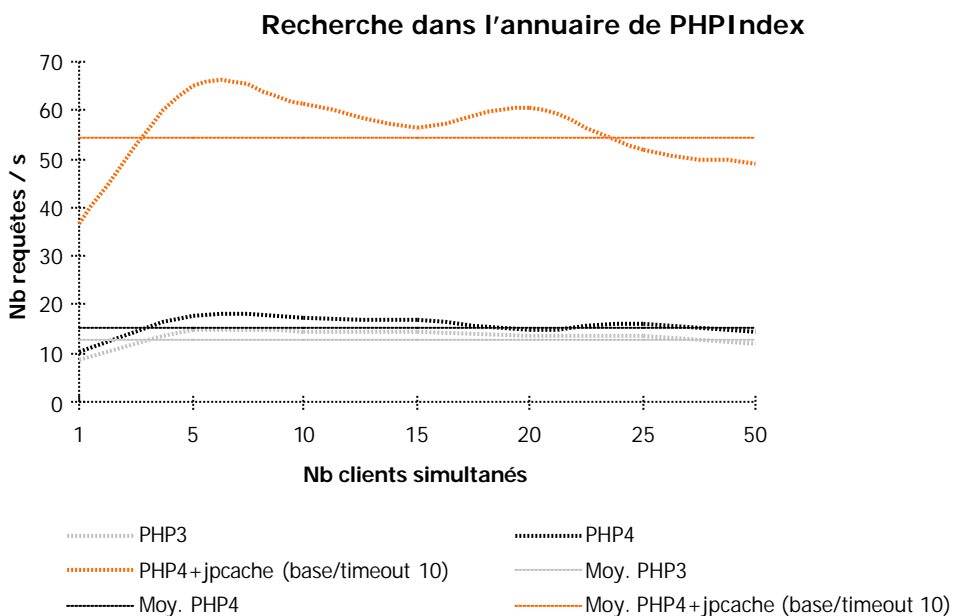
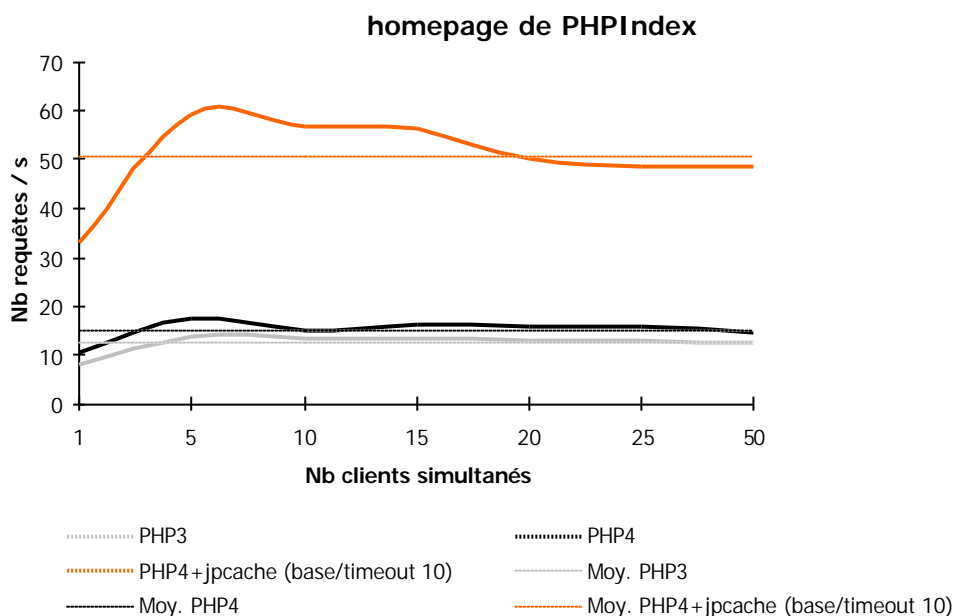
CL	CR	T	RS	FR	TT	HT	TR
1	250	4.660	53.65	0	120000	49000	25.75
5	250	4.266	58.60	0	120000	49000	28.13
10	250	6.048	41.34	0	122880	50176	20.32
15	250	7.891	31.68	0	120960	49392	15.33
20	250	9.910	25.23	0	121440	49588	12.25
25	250	11.785	21.21	0	120480	49196	10.22
50	250	17.897	13.97	0	122400	49980	6.84

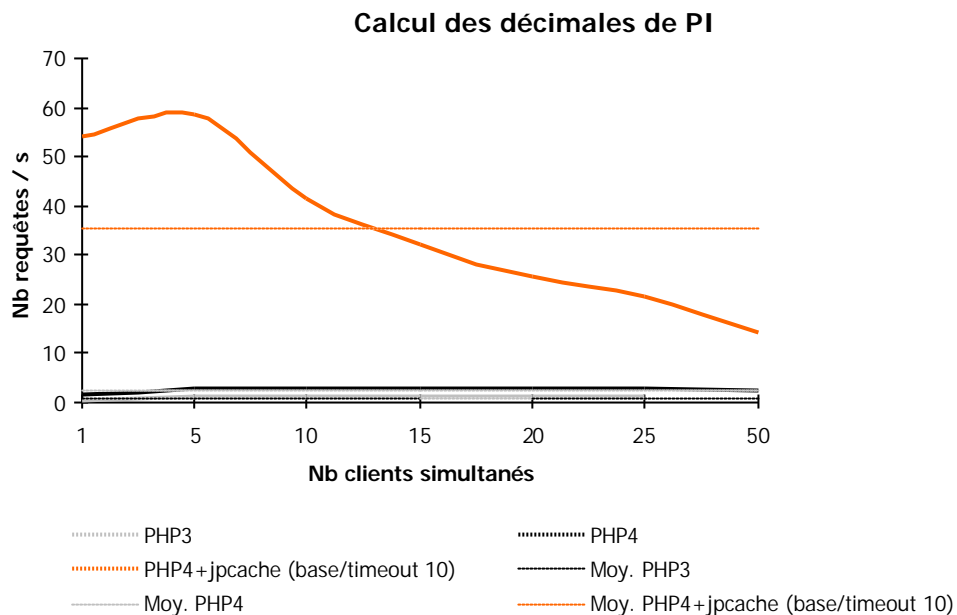
16.2 Configuration

PHP4.0.6 + jpcache 1.1.1 (jpcache-sql.php)

\$cachetimeout=10

16.3 Analyse





Les résultats obtenus restent bons ! Et ceci pour l'ensemble des scripts.

A noter tout de même un écroulement des performances sur le dernier script lorsque le nombre de clients simultanés augmente.

17. PHP4+ZendCache+ZendOptimizer+jpcache (fichier / timeout 10)

17.1 Résultats des mesures

17.1.1 Script A : homepage de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	4.420	56.56	0	1251500	1188250	283.14
5	250	2.573	97.16	0	1261512	1197756	490.29
10	250	2.620	95.42	0	1290296	1224769	492.48
15	250	2.648	94.41	0	1283204	1217930	484.59
20	250	2.694	92.80	0	1304896	1238104	484.37
25	250	2.722	91.84	0	1318246	1251201	484.29
50	250	2.816	88.78	0	1289044	1223264	457.76

17.1.2 Script B : recherche dans l'annuaire de PHPIndex

CL	CR	T	RS	FR	TT	HT	TR
1	250	4.501	55.54	0	1141750	1078000	253.67
5	250	2.556	97.81	0	1155077	1090307	451.91
10	250	2.576	97.05	0	1173345	1107555	455.49
15	250	2.623	95.31	0	1182853	1116808	450.95
20	250	2.638	94.77	0	1182105	1115550	448.11
25	250	2.697	92.70	0	1189966	1123156	441.22
50	250	2.807	89.06	0	1200373	1132798	427.64

17.1.3 Script C : calcul des décimales de PI

CL	CR	T	RS	FR	TT	HT	TR
1	250	2.870	87.11	0	112500	49000	39.20
5	250	1.821	137.29	0	112950	49196	62.03
10	250	1.801	138.81	0	113850	49588	63.21
15	250	1.832	136.46	0	114750	49980	62.64
20	250	1.854	134.84	0	117450	51156	63.35
25	250	1.843	135.65	0	115200	50176	62.51
50	250	1.929	129.60	0	114300	49784	59.25

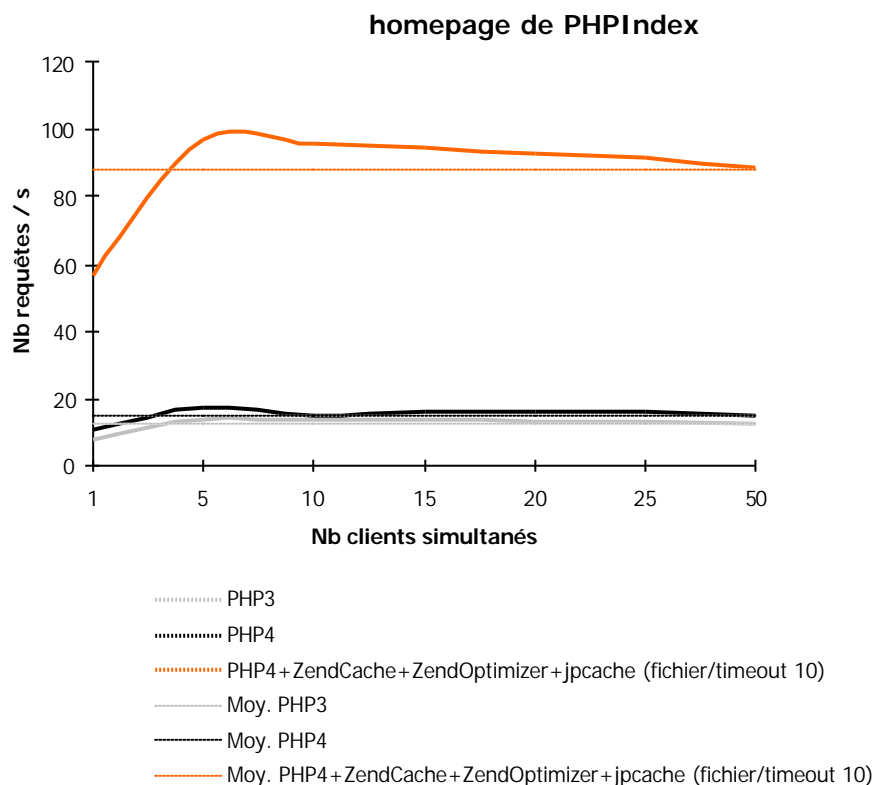
17.2 Configuration

PHP4.0.6 + ZendOptimizer-1.1.0 + ZendCache-1.1.0b (Linux_glibc2.1) + jpcache 1.1.1 (jpcache.inc)

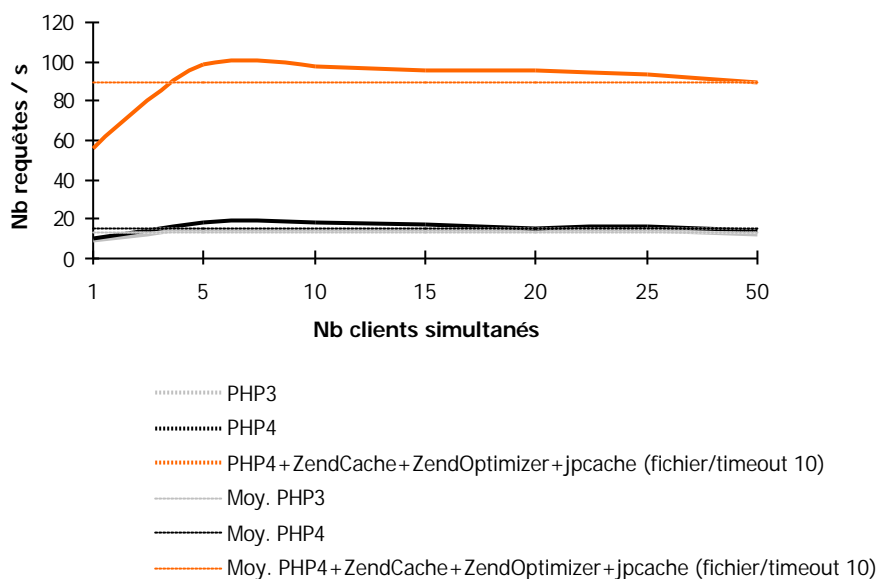
```
zend_optimizer.optimization_level=15  
zend_extension=/usr/local/Zend/lib/ZendOptimizer.so  
zend_cache.memory_consumption=64  
zend_cache.validate_timestamps=1  
zend_cache.use_cwd=1  
zend_extension=/usr/local/Zend/lib/ZendCache.so
```

```
$cachetimeout=10
```

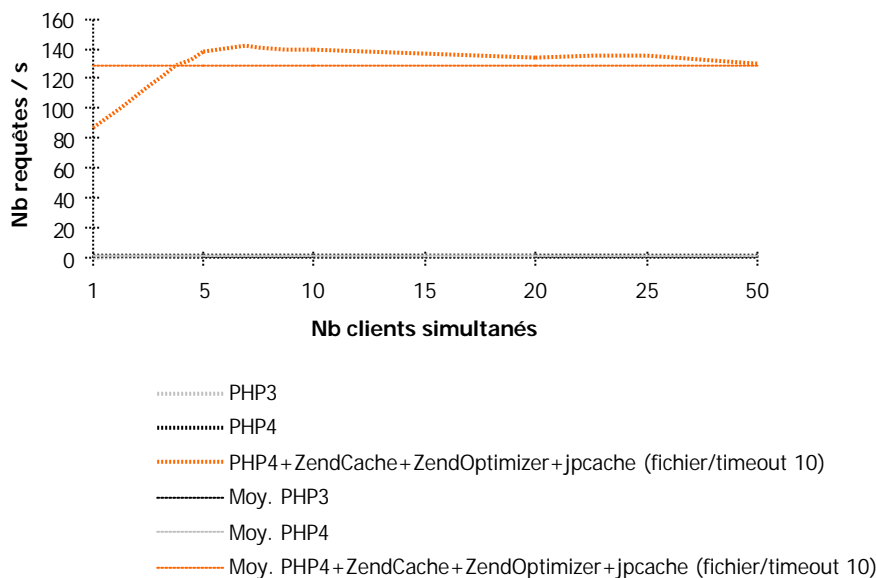
17.3 Analyse



Recherche dans l'annuaire de PHPIindex



Calcul des décimales de PI



Il semblait intéressant de combiner l'ensemble des solutions ayant permis d'obtenir de bonnes performances.

A vrai dire, la combinaison est gagnante et les résultats sont les meilleurs pour tous les scripts.

18. Conclusion

Les résultats observés démontrent clairement l'apport majeur du passage à PHP4 par rapport à PHP3. Mais aussi le fait que les différentes solutions étudiées sont complémentaires.

Nous avons vu aussi que l'Alternative PHP Cache est dans tous les cas moins performant que son concurrent de chez Zend. De plus, il tend à charger énormément le serveur en CPU et mémoire. Mais il présente aussi l'avantage d'être sous une licence libre. Et il possède des fonctionnalités intéressantes que ne propose pas Zend Cache. En particulier la possibilité de pouvoir paramétrer différemment l'action du cache d' « opcode » suivant tel ou tel script.

Le Zend Optimizer est performant dans des cas bien particuliers, tel un script de calcul pur. Mais ceci est loin d'être représentatif d'une mise en œuvre typique des technologies Web dynamique. Cependant, le Zend Optimizer a sa raison d'être. Et le fait qu'il soit librement téléchargeable aussi. En effet, le Zend Optimizer est nécessaire à l'exécution d'un autre produit de Zend : le Zend Encodeur.

Quant au jpcache, il se montre très performant, surtout en mode fichier. Il est également tout à fait adapté à une utilisation en environnement mutualisé, ne nécessitant pas le paramétrage du php.ini ou le redémarrage du serveur. Cependant, il n'est pas adapté à l'ensemble des problématiques (page fréquemment mise à jour, INSERT ou UPDATE, sessions, etc.). Il implique également un minimum de travail de la part du ou des développeurs afin d'affiner les paramétrages de mise en cache. Et son intégration peut s'avérer complexe dans le cas d'un projet en production.

Dans ce cas, l'utilisation de la combinaison Zend Cache/Zend Optimizer apporte une plus grande rapidité d'exécution que PHP3 ou PHP4 seul. Mais elle nécessite un environnement dédié ou les bonnes dispositions du ou des administrateurs systèmes, même si une solution encore en test à ce jour pourrait simplifier la chose : le Zend Cache Shared Server.

	homepage	Recherche dans l'annuaire	Calcul des décimales de PI
PHP4	28%	22%	199%
PHP4+ZendOptimizer	25%	21%	346%
PHP4+ZendCache	51%	28%	199%
PHP4+ZendCache+ZendOptimizer	53%	28%	347%
PHP4+APC SHM	36%	24%	197%
PHP4+APC MMAP	33%	24%	197%
PHP4+jpcache (fichier/timeout 900)	531%	543%	17197%
PHP4+jpcache (fichier/timeout 10)	536%	516%	11712%
PHP4+jpcache (base/timeout 900)	349%	345%	12749%
PHP4+jpcache (base/timeout 10)	333%	343%	6484%
PHP4+ZendCache+ZendOptimizer+jpcache (fichier/timeout 10)	608%	568%	15326%

Gains de performances par rapport à PHP3

	Licence	Coût d'achat
Zend Optimizer	Zend Optimizer	Librement téléchargeable
Zend Cache	Zend Cache	Entre 1875 et 9000 US\$ par processeur selon le type de celui-ci. Prix dégressif.
APC cache	QPL	Librement téléchargeable
jpcache	GPL	Librement téléchargeable

Récapitulatif des licences et des coûts

A Vers la standardisation d'un benchmark Web

L'étude effectuée ici pourrait largement être étendue et approfondie. Et ceci pour de multiples raisons :

- L'outil de mesure utilisé, Apache Bench, est largement perfectible. A ce titre, compléter ces mesures avec d'autres outils comme httpperf ou Autobench serait intéressant.
- Si Apache est certainement le serveur HTTP le plus utilisé au monde, mesurer les répercussions de l'utilisation d'alternatives à Apache serait également judicieux. C'est en particulier le cas de IIS, le serveur Web de Microsoft. Ce dernier présente une caractéristique intéressante. A savoir, d'opter pour un modèle d'architecture dit «multi-threadé ». A la différence d'Apache qui, en attendant Apache 2, utilise un modèle «pré-fork ». Or, si ce dernier se montre plus robuste, le modèle « multi-threadé » offre quant à lui une bien meilleure réponse aux montées en charges.
- Jouer plus finement sur la puissance CPU et la mémoire eût été opportun. On constate par exemple que le Zend Cache est moins gourmand que l'APC. Des disparités existent donc aussi à ce niveau là.
- Et plus largement, chercher à étudier la charge d'autres solutions de scripting comme Perl (ou mod Perl), les ASP, les JSP ou encore Cold Fusion serait également intéressant.

Le sujet apparaît donc rapidement comme très vaste, tant la diversité des technologies, des solutions logicielles et matérielles, sont grandes.

Il serait néanmoins vraiment utile de chercher à standardiser tout cela. Et si la tâche semble délicate, elle n'est pas impossible. Un bon début consisterait sans doute par définir une architecture logicielle et matérielle de référence au niveau serveur. Et ceci tant sous Apache que IIS. Puis réfléchir à l'élaboration d'une collection de scripts portables (en particulier d'une solution de scripting à une autre) et mixant calcul, lecture et insertion dans un SGBD, recherche, etc. Enfin, définir les conditions des tests (combien de requêtes, combien de clients, combien de temps, etc.).

Il serait alors beaucoup plus simple de comparer les performances de telle ou telle solution par rapport à une autre : par exemple Apache/PHP/MySQL d'un coté et IIS/ASP/SQLServer de l'autre. Puis mesurer les évolutions en utilisant PostgreSQL au lieu de MySQL.

En fait, le but ultime ne serait-il pas tout simplement de pouvoir disposer d'un équivalent Web des célèbres SpecInt et SpecFloat du monde des processeurs ? Ce serait en tout cas fort judicieux.

B A propos de GLOBALIS media systems

GLOBALIS media systems, spécialiste technique Web, offre à ses clients, web agencies, dotcom ou grands comptes, une expertise et un développement technique de haut niveau.

Développement technique

- Développement de sites fortement dynamiques.
- Commerce électronique.
- PHP4, PHP3, ASP, Perl.
- HTML, DHTML, XML, WML.
- MySQL, PostgreSQL, Oracle, SQLServer.

Etudes

- Audit, Conseil.
- Définition des besoins.
- Rédaction de cahiers des charges.
- Assistance à maîtrise d'ouvrage.
- Transfert de compétences.

Supports

- Internet, Intranet, Extranet.
- WAP, I-Mode, PDA.

Veille

- Linux et les Logiciels Libres.
- Internet Mobile.
- Technologies émergentes.

GLOBALIS media systems contribue également à l'essor des technologies OpenSource en publiant le site PHPINDEX <http://www.phpindex.com>, portail de veille dédié aux technologies PHP, en ouvrant le code source de certaines applications comme phpFAQTory et en participant à des projets majeurs comme phpMyAdmin.

Ses clients ont pour nom Synerdeal, CNRS, Crédit Agricole, 3COM, France Soir, Business Interactif, Artifica, MATRA Systèmes & Information, CNFPT.

Contact : Frédéric HOVART, 01.56.08.00.99, fred.hovart@globalis-ms.com.

GLOBALIS media systems
25, rue Thiboumery - 75015 Paris
Tel : +33 (0)1.56.08.00.99 - Fax : +33 (0)1.56.08.09.27
<http://www.globalis-ms.com>
infos@globalis-ms.com

C Ressources

▪ Outils de benchmark

Apachebench	Fourni avec Apache Server
Httpperf	http://www.hpl.hp.com/personal/David_Mosberger/httpperf.html
Autobench	http://www.xenoclast.org/autobench/
MWAS	http://webtool.rte.microsoft.com/

▪ Logiciels utilisés au niveau serveur

Slackware	http://www.slackware.org/
Apache Server	http://httpd.apache.org/
MySQL	http://www.mysql.com/
PHP	http://www.php.net/

▪ Logiciels testés

Zend	http://www.zend.com
Zend Optimizer	http://www.zend.com/store/products/zend-optimizer.php
Zend Cache	http://www.zend.com/store/products/zend-cache.php
Alternative PHP Cache	http://apc.communityconnect.com/
Jpcache	http://www.weirdpier.com/jpcache/
License Zend Optimizer	http://www.zend.com/store/products/optimizer-agreement.php
License Zend Cache	http://www.zend.com/store/products/cache-agreement.php
License GPL	http://www.phpindex.com/projets/projets_gpl.php3

▪ Différents benchmarks déjà réalisés

<http://www.reviewboard.com/Section/Cover/ZendCache>
http://www.zend.com/store/pdf/Cache_Benchmark_30April01.pdf
<http://www.manucorp.com/article.php3?id=12>

▪ Textes de références

<http://www.usenix.org/publications/library/proceedings/usits97/banga.html>

D Le script de calcul des décimales de PI

```
<?php
//
// +-----+
// | Calcul des décimales de PI |
// +-----+
// | Copyright (c) 1999-2001 The PHPIndex Group |
// +-----+
// | This program is an open source software; you can redistribute it |
// | and/or modify it under the terms of one of the GNU General Public |
// | License as published by the Free Software Foundation; either |
// | version 2 of the License, or (at your option) any later version. |
// +-----+
// | Authors: |
// | Julien Wajsberg <flash@minet.net> |
// | Armel Fauveau <armel.fauveau@globalis-ms.com> |
// +-----+
// | $RCSfile: pi.php,v $ |
// | $Author: armel $ |
// | $Revision: 1.1.1.2 $ |
// | $Date: 2001/08/25 20:27:49 $ |
// +-----+

// {{{ définition de variables
define(NB_ITERATION,10000);
define(PRECISION,48);
// }}}
// {{{ calcul
echo "Calcul des d&eacute;cimales de PI<br>\n";
echo "Nombre d'it&eacute;rations : ".NB_ITERATION."<br>\n";
echo "<br>\n";
echo "Calcul des suites Un et Vn...<br>\n";

$u[1] = 99./100;
$u[2] = 4801./5000;
$v[1] = 99./4780;
$v[2] = -11414399./11424200;

for ($i = 3 ; $i <= 2*NB_ITERATION-1 ; $i++) {
    $u[$i] = 99./50 * $u[$i-1] - $u[$i-2];
    $v[$i] = 99./2390 * $v[$i-1] - $v[$i-2];
}
echo "Termin&eacute;.<br>\n";
echo "<br>\n";
echo "Calcul du nombre PI...<br>\n";

$sum = 0;
for ($i = 1 ; $i <= NB_ITERATION ; $i ++ ) {
    $sum += pow(-1, $i-1)/(pow(10,2*$i-1)*(2*$i-1)) * (4*$u[2*$i-1] - $v[2*$i-1]);
}
echo "Termin&eacute;.<br>\n";
echo "<br>\n";
echo ("Le nombre PI calcul&eacute; :<br>\n".number_format(8 * $sum,
PRECISION)."<br>\n");
echo ("Le nombre PI fourni par PHP :<br>\n".number_format(M_PI,
PRECISION)."<br>\n");
// }}}
?>
```