clever age

conseil en architecture technique

Etude de solutions d'optimisation pour PHP

Publication : Octobre 2001

Auteur : Matthieu Belge

<mbelge@clever-age.com>

SOMMAIRE

1 - Objectifs et méthodologie

Cette étude a pour objectif de déterminer l'apport en terme de performance des trois principales solutions de cache pour PHP4 :

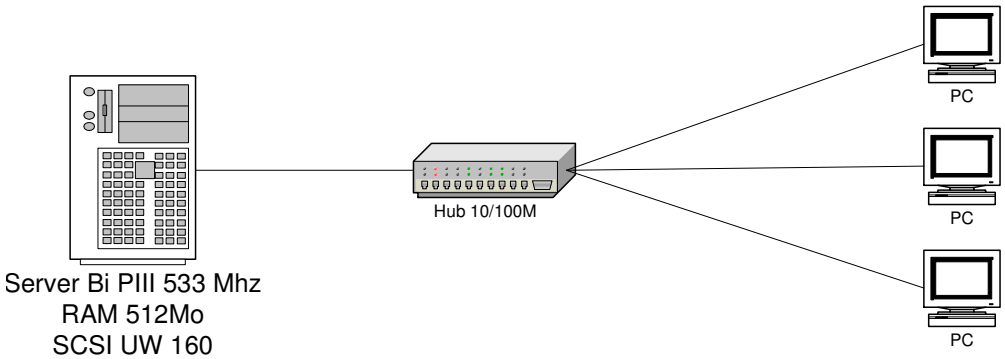
- Zend Cache 1.1.0, solution commerciale de Zend Technologies
- Alternative PHP Cache (APC) 1.1.0
- Bware 0.10, connu aussi sous le nom de afterBURNER*Cache.

Ces deux dernières sont des solutions libres.

Le principe de fonctionnement de ces caches réside dans le fait que les scripts sont compilés et stockés dans la mémoire du serveur. On évite donc des accès disques et compilations systématiques à chaque appel d'un script PHP. Il s'agit bien d'un cache de compilation car l'exécution des scripts est effectuée à chaque appel. Le fonctionnement des applications n'est donc pas altéré par ce cache.

Le deuxième objectif de cette étude est de déterminer quelle solution de cache est la mieux adaptée aux différentes types de sites Web.

Afin d'évaluer les performances de chacune, nous avons réalisé un test de montée en charge sur l'architecture suivante :



Les PC postes de tir ont été équipés d'injecteurs HTTP qui génèrent une charge croissante jusqu'à 100 automates. Les automates produisent une charge « stressante » puisque aucun temps d'attente entre les pages n'a été programmé.

L'application testée est PHPNuke 4.4.1a[1], un outil de publication pour le web largement répandu, se servant de MySQL comme base de données. L'application a été alimenté par une centaine d'articles et une vingtaine d'utilisateurs.

Le scénario de test consiste à naviguer sur 4 pages du site utilisant PHPNuke :

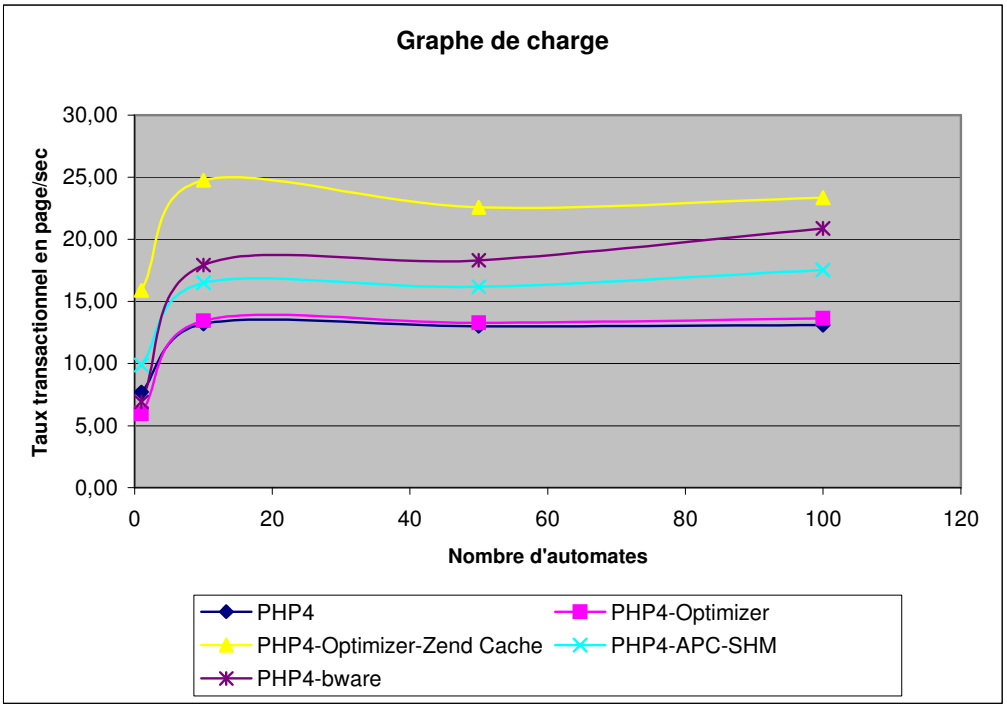
- Page d'accueil : Mise en page et affichage des dix derniers articles.
- Une recherche : Mise en page et affichage des dix premiers résultats.
- Lecture d'un article : Mise en page et affichage d'un article.
- Lecture d'un deuxième article : Mise en page et affichage d'un article.

Durant les tirs, on relève alors les temps de réponse de chacune des pages afin de déterminer le taux transactionnel (capacité du serveur à délivrer des pages chaque seconde) de chaque solution. Les consommations en ressources système (CPU et RAM) sont également suivies. Cette observation permet d'identifier les éventuels goulets d'étranglement liés a l'utilisation de ces caches. La colonne gain dans les tableaux de résultats corresponds au gain de performances par page.

[1] PHPNuke : http://www.phpnuke.org/

2 - Synthèse des résultats

2.1 - Performances



Ce graphe de synthèse montre incontestablement le gain de performance qu'apportent les diverses solutions de cache sur le taux transactionnel. La solution Zend Cache obtient le meilleur gain de performance en moyenne avec 180% environ par rapport à APC et afterBURNER*Cache (Bware) obtenant respectivement 130% et 135%.

Il faut aussi noter que même lors de faible charge le temps de latence de trouve fortement réduit (cf. 1 automate).

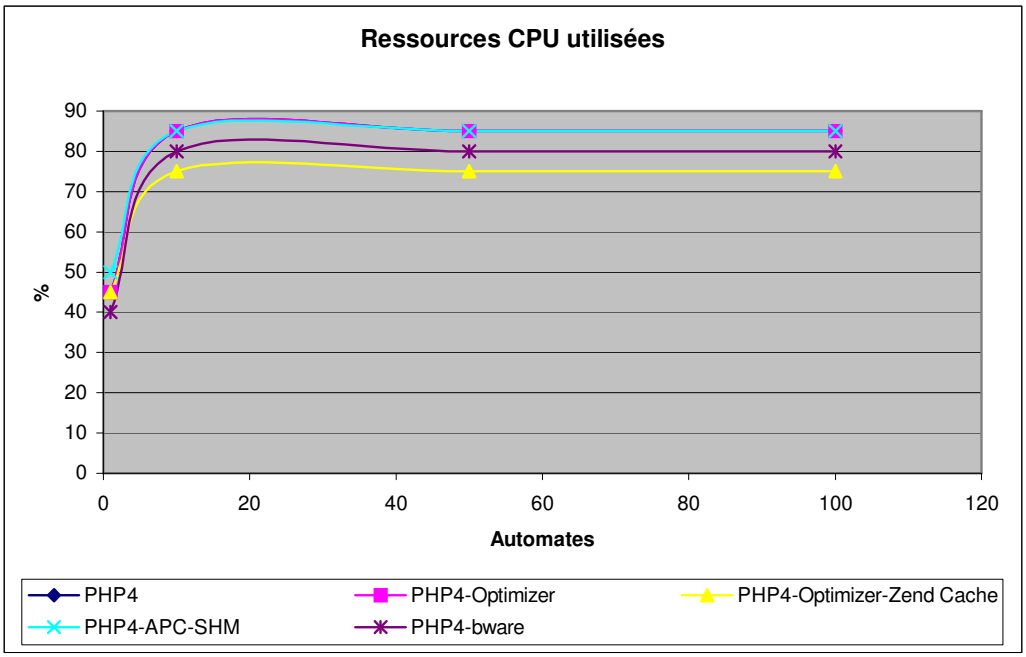
Pages	Zend Cache	APC	AfterBURNER*Cache
Article1	300%	156%	218%
Article2	302%	153%	220%
Index	140%	119%	135%
Search	137%	128%	133%

Le tableau ci-dessus présente les gains de performance par page pour chaque solution. On remarque que la lecture d'un article est plus efficace que la page d'accueil et la page de recherche. En effet, ces deux dernières pages effectuent

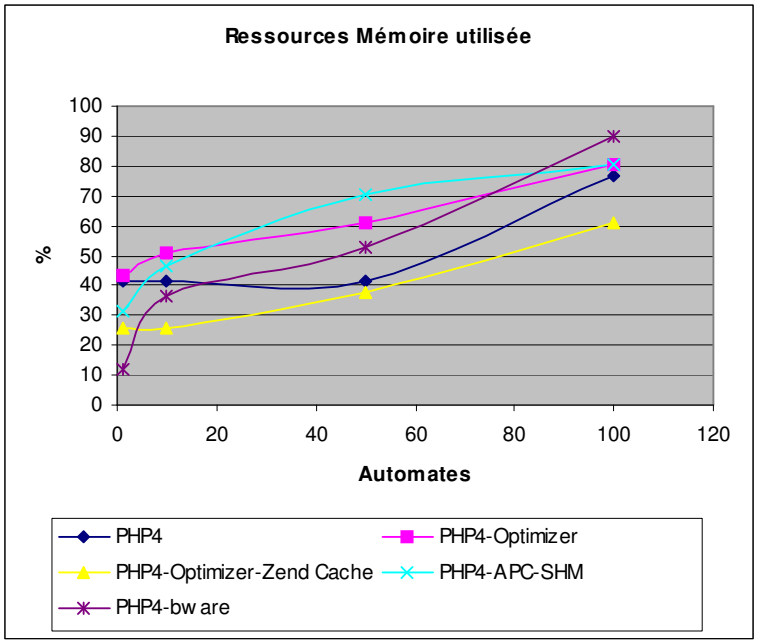
une requête vers la base MySQL plus longue à exécuter (récupération des 10 derniers articles). Ainsi, lors de ces tests, les performances constatées sont plus liées au temps d'exécution des requêtes qu'à l'interprétation du PHP proprement dit.

2.2 - Ressources système

Un des points secondaires les plus importants, finalement, dans ces tests est certainement l'économie, au niveau des ressources systèmes, que peuvent apporter les solutions de cache. En effet, cette économie est due à la phase de compilation qui n'est faite qu'une fois lors du premier appel des scripts. Ces résultats ne sont pas surprenant, mais dans le cas du Zend Cache les gains de ressources mémoires sont plutôt une surprise inattendue.



Le graphe ci-dessus représente, pour chaque solution de cache, les ressources CPU utilisées en fonction de la montée en charge. L'écart se creuse en fonction de la charge. On remarque ainsi que le Zend Cache est le moins consommateur en ressource CPU (75% utilisé). Il est suivi par l'afterBURNER*Cache (80%). Les autres solutions sont à égalité (85%).



Le graphe ci-dessus représente, pour chaque solution de cache, les ressources mémoire utilisées en fonction de la montée en charge. On remarque bien les grandes différences des différentes solutions de cache. En effet, le Zend Cache apporte indéniablement une réelle économie de ressource, alors que les autres solutions sont plutôt consommatrices de ressource mémoire. Nous avons donc en moyenne, dans l'ordre, avec une charge de 100 automates : Zend Cache (60%), APC (80%), afterBURNER*CACHE (90%).

3 - Conclusions de l'étude

Les solutions de cache ont donc plusieurs avantages :

- Amélioration du taux transactionnel du serveur, c'est à dire sa capacité à délivrer des pages. Possibilité donc d'accueillir une population d'internautes plus importante sur un serveur.
- Réduction du temps de latence (grâce à la suppression de la phase de compilation) qui est perceptible même à faible charge. Les solutions de cache ont donc un intérêt pour les sites peu chargés mais qui souhaitent réduire les temps d'attente de leurs utilisateurs.
- Optimisation des ressources du serveur. En plus d'accroître le taux transactionnel, certaines solutions de cache (Zend Cache et afterBURNER*CACHE) limitent la consommation en ressource. Raison de plus pour ne pas s'en priver.

Clever Age recommande l'usage d'un cache PHP dans les conditions suivantes :

- Site à faible ou moyenne audience qui aimerait réduire les temps d'attente des utilisateurs.
- Site à large audience afin d'optimiser l'utilisation des serveurs et éviter l'ajout de serveurs supplémentaires.
- Sites sur serveur mutualisé ce qui permet d'accroître leur capacité d'accueil.

Le choix entre chacune des solutions doit s'appuyer sur les critères suivants :

- Performances (taux transactionnel, consommation CPU et RAM)
- Le budget (seul Zend Cache est payant avec un ticket d'entrée minimum de 800 Euros par an).
- Support technique et assistance de l'éditeur (apporté uniquement par Zend)

De manière générale et au delà des tests réalisés pour cette étude, nous avons constaté des gains de performance sur des sites en production de 200 à 600%.

Il faut néanmoins garder à l'esprit que la nature des développements et les traitements externes à PHP (processeur XSLT, procédures stockées en base …) peuvent influer fortement sur les gains constatés.

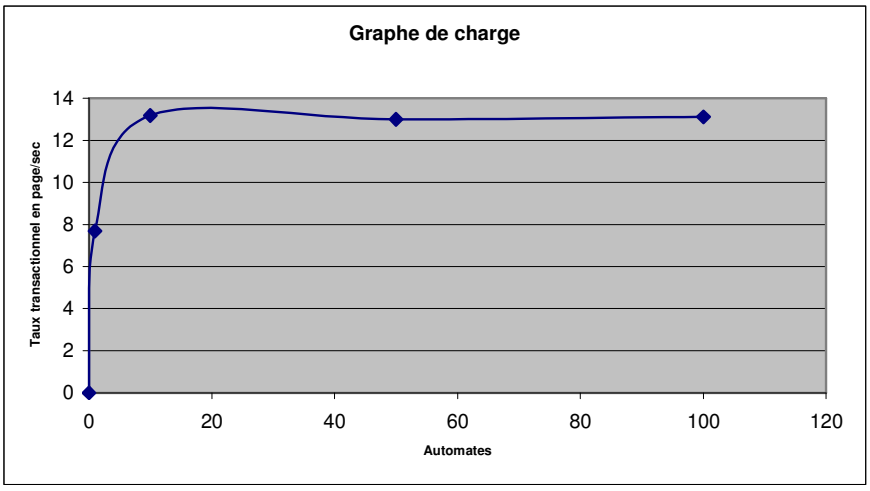
4 - Résultats détaillés

4.1 - PHP4 seul

Le tableau ci-dessous résume les principaux résultats obtenus avec PHP4 :

Nombre d'automates	1	10	50	100
Nombre de transactions	200	200	100	30
Article1	0,11	0,61	3,13	6,15
Article2	0,11	0,64	3,24	6,35
Index	0,19	1,08	5,54	11,12
Search	0,12	0,70	3,49	6,90
Temps de réponse moyen par page	0,13	0,76	3,85	7,63
Taux transactionnel	7,69	13,19	13,00	13,11
Fiabilité	100%	100%	100%	100%
Nombre de processus Apache	1	10	50	100
Nombre de connections MySQL	1	10	50	100

L'application et le serveur se sont montrés fiables et stables lors de la montée en charge. Cependant, les temps de réponse à forte charge (100 automates en test stressant) sont élevés (près de 8 secondes)



Le graphe de charge ci-dessus confirme la stabilité du taux transactionnel à partir de 10 automates. Les dégradations constatées par les utilisateurs, toutes choses égales par ailleurs, sont donc proportionnelles à l'accroissement du nombre total d'utilisateurs.

Nombre d'automates	1	10	50	100
CPU utilisé en %	45	85	85	85
Mémoire utilisée / 512Mo en %	41	41	41	77

Le tableau ci-dessus correspond aux consommations en ressource CPU et mémoire du serveur lors de la montée en charge. Comme nous pouvons le constater, aucune ressource n'est arrivée à saturation.

4.2 - PHP4 et Zend Optimizer 1.1.0

Dans ce test, nous avons activé l'utilisation du Zend Optimizer, une extension gratuite de PHP4, téléchargeable sur le site de Zend. L'Optimizer remplit deux fonctions principales :

- Il pré-compile les scripts PHP lors de leur exécution en effectuant des passes de compilation. En effet, comme son nom l'indique, il va optimiser certaines parties de code comme pour les langages non interprétés.
- Il décode les scripts PHP encodés par le Zend Encoder. Le Zend Encoder est un produit commercial qui permet de rendre les sources PHP illisibles (protection intellectuelle, sécurité, protection contre les modifications).

La configuration de l'Optimizer est celle par défaut :

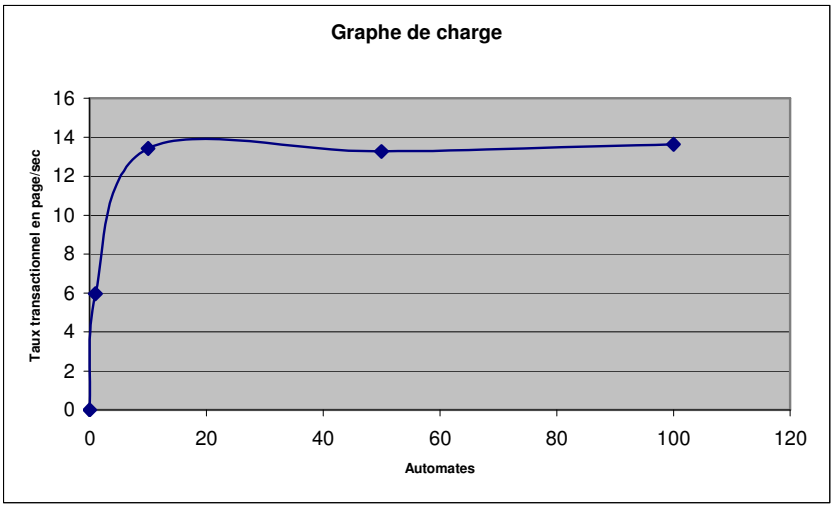
```
;Niveau de passes de compilation
zend_optimizer.optimization_level=15
;Chargement de l'extension
zend_extension="/usr/local/Zend/lib/ZendOptimizer.so"
```

Le tableau ci-dessous résume les principaux résultats obtenus :

Nombre d'automates	1	10	50	100
Nombre de transactions	200	200	100	30
Article1	0,11	0,63	3,21	6,15
Article2	0,11	0,64	3,24	6,31
Index	0,34	1,02	5,24	10,28
Search	0,12	0,70	3,40	6,61
Temps de réponse moyen par page	0,17	0,74	3,77	7,34
Taux transactionnel	5,96	13,43	13,26	13,63
Fiabilité	100%	100%	100%	100%
Nombre de processus Apache	1	10	50	100

Nombre de connections MySQL	1	10	50	100

Comme nous pouvons le constater, ces résultats sont pratiquement équivalents à ceux de PHP4 seul. Le gain sur la phase de pré-compilation n'est pas perceptible sur cette application. Ceci confirme les résultats obtenus par nos clients qui constate des améliorations de performances entre 0 et 20%. Les méthodes de développement ont clairement un impact sur les résultats du Zend Optimizer.



Une fois encore, le graphe de charge ci-dessus confirme la stabilité du serveur lors de la montée en charge.

Nombre d'automates	1	10	50	100
CPU utilisé en %	45	85	85	85
Mémoire utilisée / 512Mo en %	43	51	61	80

Le tableau ci-dessus correspond aux consommations en ressource CPU et mémoire du serveur lors de la montée en charge. Comme nous pouvons le constater aucune ressource n'est arrivée à saturation.

En revanche, nous remarquons l'augmentation de près de 34% de l'usage de la mémoire par rapport à celle utilisée lors des tests de PHP4, en raison de l'activation de l'Optimizer.

Il sera donc important de vérifier un éventuel gain de performance, lors de l'activation de l'Optimizer, avant son déploiement en production, dans le cas où il n'y aurait pas utilisation de l'Encoder. De manière générale, on peut obtenir suivant l'application en production entre 0 et 20% de gain de performance.

4.3 - PHP4, Zend Optimizer 1.1.0 et Zend Cache 1.1.0

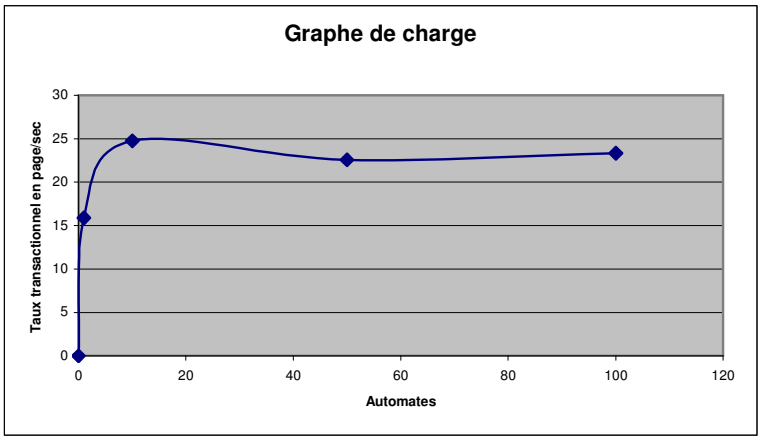
Dans ce test, nous avons activé l'utilisation du Zend Cache, un produit commercial de Zend Technologies. Le cache est téléchargeable en version d'évaluation valable 1 mois sur le site de Zend. Zend Cache est livré avec un outil de monitoring permettant d'observer les paramètres de configuration ainsi que les script les plus utilisés. En outre, un outil de benchmark basé sur Apache Bench permet d'observer le gain de performance qu'apporte le Zend Cache.

La configuration du Zend Optimizer et du Zend Cache est celle par défaut :

```
;Niveau de passes de compilation
zend_optimizer.optimization_level=15
;Chargement de l'extension
zend_extension="/usr/local/Zend/lib/ZendOptimizer.so"
;Option de verification du chemin des scripts
zend_cache.use_cwd=1
;Allocation de la mémoire pour le Cache
zend_cache.memory_consumption=64 ; 64MB
;Option de verification de la date de modification des
scripts
zend_cache.validate_timestamps=1
;Chargement de l'extension
zend_extension="/usr/local/Zend/lib/ZendCache.so"
```

Nombre d'automates	1	10	50	100	
Nombre de transactions	200	200	100	30	*Gain*
Article1	0,03	0,19	1,18	2,05	300%
Article2	0,03	0,19	1,18	2,10	302%
Index	0,13	0,83	3,95	7,94	140%
Search	0,06	0,41	2,56	5,05	137%
Temps de réponse moyen par page	0,06	0,40	2,22	4,29	
Taux transactionnel	15,87	24,75	22,56	23,34	
Fiabilité	100%	100%	100%	100%	
Nombre de processes Apache	1	10	50	100	
Nombre de connections MySQL	1	10	50	100	

Le gain de performance obtenu par l'activation du Zend Cache apparaît indéniablement. En effet, par rapport aux deux tests précédents, nous obtenons 180% de gain de performance.



Nombre d'automates	1	10	50	100
CPU utilisé en %	45	75	75	75
Mémoire utilisée / 512Mo en %	26	26	38	61

Le tableau ci-dessus correspond aux consommations en ressource CPU et mémoire du serveur lors de la montée en charge. Comme nous pouvons le constater aucune ressource n'est arrivée à saturation.

En comparant avec les résultats des ressources consommées précédemment, on observe nettement l'économie en ressource CPU et mémoire (respectivement 13% et 25% en moyenne) provoquée par l'activation du Zend Cache.

Un test complémentaire avec uniquement le Zend Cache n'a pas été effectué dans le cadre de cette étude. On peut néanmoins supposer qu'un tel test donnerait des résultats encore meilleurs sur l'économie des ressources du serveur.

4.4 - PHP4 et Alternative PHP Cache 1.1.0 en mode SHM

Dans ce test nous avons activé l'Alternative PHP Cache (APC), après avoir pris soin de désactiver le Zend Cache et le Zend Optimizer. APC fonctionne en deux modes :

- SHM : Shared Memory.
- MMAP : Memory Mapped File.

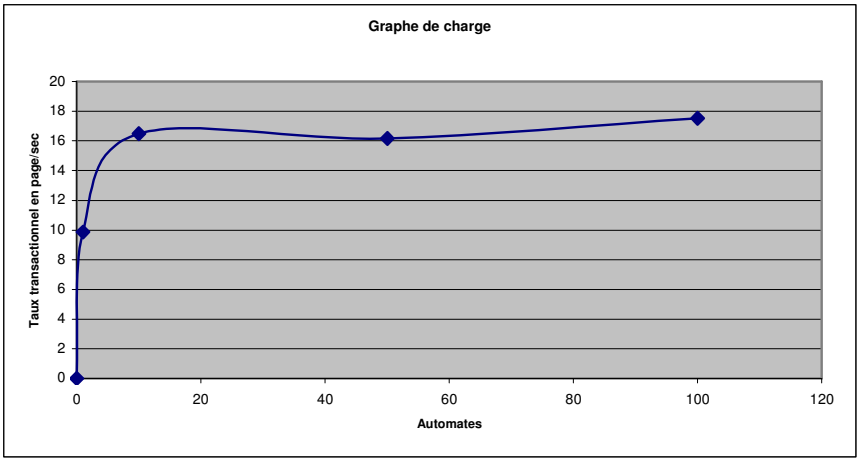
Nous n'avons testé que le mode SHM, étant donné que le mode MMAP a posé des problèmes de fiabilité au niveau du serveur.

La configuration de APC est celle par défaut :

```
;Chargement de l'extension
zend_extension=/usr/local/Zend/lib/php_apc.so
;Selection du mode de fonctionnement
apc.mode = shm
```

Nombre d'automates	1	10	50	100	
Nombre de transactions	200	200	100	30	*Gain*
Article1	0,07	0,41	2,17	3,93	156%
Article2	0,07	0,44	2,25	4,15	153%
Index	0,17	0,99	5,13	9,34	119%
Search	0,09	0,58	2,83	5,39	128%
Temps de réponse moyen par page	0,10	0,61	3,09	5,70	
Taux transactionnel	9,88	16,49	16,17	17,53	
Fiabilité	100%	100%	100%	100%	
Nombre de processes Apache	1	10	50	100	
Nombre de connections MySQL	1	10	50	100	

On peut aisément remarquer le gain de performance de 130% apporté par l'activation de l'APC.



Nombre d'automates	1	10	50	100
CPU utilisé en %	50	85	85	85
Mémoire utilisée / 512Mo en %	32	46	71	80

Le tableau ci-dessus correspond aux consommations en ressources CPU et mémoire du serveur lors de la montée en charge. Comme nous pouvons le constater aucune ressource n'est arrivée à saturation.

La consommation CPU ne diminue pas par rapport aux tests PHP4 seul, en revanche on note une augmentation de 15% de la mémoire utilisée.

4.5 - PHP4 et Bware-afterBURNER*Cache 0.10

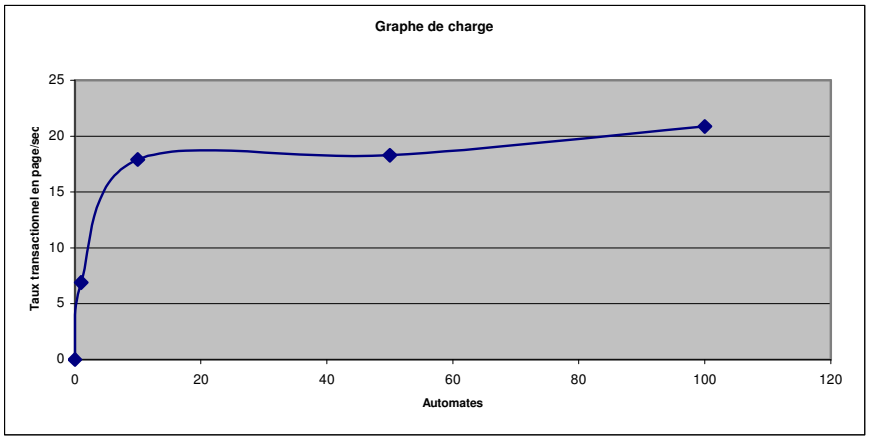
Dans ce test, nous avons activé le Cache de Bware, après avoir pris soin de désactiver les autres caches ainsi que le Zend Optimizer. Une particularité de Bware Cache est qu'il peut aussi mettre en cache des fichiers de différents types (*.tpl, *.html,…).

La configuration de Bware Cache est celle par défaut :

```
;Chargement de l'extension
zend_extension="/usr/local/lib/bware_cache.so"
;Activation du cache
bware_cache_enable="1"
;Verification de la modification du fichier
bware_cache_check_stamp="1"
;Activation des logs de Bware
bware_cache_log_level="0"
;Selection des fichiers a cacher
bware_cache_only="*.php *.tpl"
```

Nombre d'automates	1	10	50	100	
Nombre de transactions	200	200	100	30	*Gain*
Article1	0,07	0,35	1,71	2,82	218%
Article2	0,07	0,36	1,76	2,89	220%
Index	0,35	0,98	4,88	8,26	135%
Search	0,09	0,54	2,58	5,20	133%
Temps de réponse moyen par page	0,14	0,56	2,73	4,79	
Taux transactionnel	6,91	17,92	18,29	20,87	
Fiabilité	100%	100%	100%	100%	
Nombre de processes Apache	1	10	50	100	
Nombre de connections MySQL	1	10	50	100	

De même façon que pour les autres solutions, l'utilisation de Bware Cache apporte une amélioration des performances de 135%.



Nombre d'automates	1	10	50	100
CPU utilisé en %	40	80	80	80
Mémoire utilisée / 512Mo en %	12	37	53	90

Le tableau ci-dessus correspond aux consommations en ressources CPU et mémoire du serveur lors de la montée en charge. Comme nous pouvons le constater aucune ressource n'est arrivée à saturation.

En comparaison avec les tests de PHP4 seul, on observe une économie de 5% des ressources CPU, mais une augmentation de 15% de la mémoire utilisée.

Clever Age

La mission de Clever Age est d'aider les entreprises à construire des architectures informatiques efficaces. Nos consultants apportent une connaissance approfondie des solutions du marché, doublée d'une forte expérience terrain. Cette double compétence assure des choix adéquats au système d'information existant.

- Résister aux modes technologiques ("hype")
- Décrypter et analyser les véritables capacités techniques et fonctionnelles des produits
- Anticiper les difficultés de mise en œuvre dans un contexte client
- Choisir la solution offrant un retour sur investissement optimal (développement collaboratif, achat de solution packagée, développement spécifique, ...)
- Assurer le pilotage de projets de manière indépendante

Tels sont nos engagements.

Pour assumer ce rôle pleinement, nous concentrons nos efforts sur les architectures basées sur Java/J2EE, Zend/PHP et Microsoft .Net.

Siège social : 54 Boulevard de Grenelle
75015 Paris

Bureaux : Parc technologique
Bâtiment Carnot - Hall 10
18/20, avenue Edouard Herriot
92350 Le Plessis Robinson

Téléphone : 01 40 83 34 47

Fax : 01 40 83 34 30

Site web : http://www.clever-age.com

Informations commerciales : commercial@clever-age.com

Informations techniques : tech@clever-age.com

AVERTISSEMENT : Droit de propriété intellectuelle

Art. L 335 - 2 :

Toute édition d'écrits, de composition musicale, de dessin de peinture ou de toute autre production, imprimée ou gravée en entier ou en partie, au mépris des lois et règlements relatifs à la propriété des auteurs, est une contrefaçon; et toute contrefaçon est un délit.

La contrefaçon en France d'ouvrages publiés en France ou à l'étranger est punie de deux ans d'emprisonnement et de 1.000.000 F d'amende (L. n° 94-102, 5 fév. 1994, art. 1er)

Art. L 335 - 8 :

Les personnes morales peuvent être déclarées responsables pénalement dans les conditions prévues par l' article 121 - 2 du Code Pénal des infractions définies aux articles L 335-2 à L 335-4 du présent code.