Impact des robots d'indexation sur le cache de second niveau de SPIP

IMBERTI Christophe - SG/SPSSI/CP2I/DO Ouest

06/06/2012 – mis à jour le 05/07/2012

SOMMAIRE

Je remercie les membres de la communauté SPIP pour leurs apports sur ce sujet.

1. Le constat

Plusieurs messages sur les listes de discussion de la communauté SPIP indiquent que **la taille du cache de second niveau de SPIP 2 est conséquente**.

Le record semblait avoir été atteint sur le site Agoravox.fr, qui comprend 50 000 articles et 1 000 000 de commentaires et dont le cache a atteint la taille de 2,7 Go. Depuis, un autre site, avec 16 000 articles, a atteint la taille de 10 Go de cache (sans application de quota de cache).

2. L'explication

Les responsables de ce phénomène **sont les robots d'indexation** des moteurs de recherche (ainsi que les aspirateurs de site) qui parcourent toutes les pages du site, ce qui génère, pour chaque page consultée, plusieurs fichiers dans le cache de second niveau de SPIP 2.

A noter que si l'on vide le cache de SPIP, il se remplira à nouveau lors des prochains passages des robots d'indexation.

La taille d'un fichier dans le cache de second niveau de SPIP 2 est généralement faible. Aussi, le phénomène constaté concerne la prolifération du nombre de fichiers dans le cache de second niveau de SPIP. Par ailleurs, le nombre de sous répertoires du cache de second niveau de SPIP 2 est de 16.

Par conséquent, le phénomène observé peut conduire à différentes situations :

- Une **saturation du nombre d'inodes** utilisés (chaque fichier nécessite un inode).
- Un très grand nombre de fichiers dans un même répertoire (cela peut provoquer une baisse des performances, qui dépend du « FileSystem » utilisé, et peut **ralentir** certaines fonctions de SPIP).
- Une **saturation de l'espace disque**.

3. Les solutions

3.1 Solution native de SPIP

SPIP 2 dispose d'un mécanisme permettant, via une tâche de fond, de supprimer une partie des fichiers en cache, lorsque la taille du cache dépasse le quota (fixé par défaut à 10 Mo).

Toutefois, d'après la communauté SPIP, ce mécanisme à plusieurs limites :

- La taille d'un répertoire du cache n'est pas mesurée mais estimée (pour des questions de performances) sur la base de 20 fichiers pris au hasard.
- « Chaque fois que ce mécanisme est appelé, il supprime jusqu'à 6% du cache, et uniquement les fichiers qui n'ont pas servi récemment. Il est donc possible que la taille du cache reste supérieure au quota. »
- « Par ailleurs, si le nombre de fichiers est trop élevé, le mécanisme n'arrive même plus à compter la taille du cache ni à le vider, si le serveur est trop lent. »

Plusieurs messages sur les listes de discussion de la communauté SPIP montrent que **cette solution ne suffit pas**.

Solution	Avantages	Inconvénients
Solution native de SPIP de limitation de la taille du cache	Limite la taille du cache de SPIP à un quota que l'on peut fixer.	« La taille réelle peut être supérieure au quota ». « Si le nombre de fichiers est trop élevé, le mécanisme n'arrive même plus à vider le cache, si le serveur est trop lent ». Si le quota est trop faible, le passage du robot provoquera le calcul de nombreuses pages.

3.2 Solution du plugin Mémoïzation

Le plugin Mémoïzation (http://plugins.spip.net/memoization.html) offre plusieurs méthodes de gestion du cache de SPIP :

- filecache,
- xcache (si le serveur dispose du cache d'opcode xcache),
- apc (si le serveur dispose du cache d'opcode apc),
- eaccelerator (si le serveur dispose du cache d'opcode eaccelerator),
- memcache (si le module PHP additionnel memcache est installé. Il permet de stocker des variables ou des pages entières dans la mémoire vive du serveur),
- nocache (pas de cache).

Le plugin **surcharge** les fichiers public/cacher.php et action/purger.php de SPIP.

3.2.1 Méthode « filecache »

Citation d'un développeur du noyau de SPIP :
« Le plugin mémoization propose une version alternative de la gestion du cache (méthode filecache), qui ne présente aucun des inconvénients de la méthode native de SPIP (nombre maximal de fichiers cache limité et fixé par construction, pas de purge). »

Le nombre de fichiers dans l'un des 16 sous-répertoires de /cache peut devenir très important et impacter les performances du système de fichiers du serveur.

La solution proposée par cette méthode « filecache » consiste à :

- Augmenter le nombre de sous répertoires de /cache, en passant de 16 à 256.
- Limiter le nombre maximal de fichiers dans le cache à 65 536 fichiers (16^4).

Exemple de nommage d'un fichier : tmp/cache/ab/cd

Avec cette méthode, le cache n'a pas vocation à stocker l'ensemble du site, mais seulement la partie consultée récemment.

Si un **robot d'indexation** parcourt le site, il ne pourra pas remplir le cache au-delà de 65 536 fichiers. Toutefois, lors du passage du robot, une page, qui ne figure pas déjà dans le cache de second niveau, sera calculée et stockée dans le cache, en écrasant éventuellement une autre page.

Cette méthode s'apparente à un quota sur le nombre de fichiers, avec une priorité aux dernières pages appelées.

Remarque :
Si une page est stockée dans ac/4f, puis qu'une autre page différente doit être mise dans le cache sous ac/4f, alors la seconde écrase la première. Comme le contenu du fichier de cache comprend l'identifiant de la page (et sa date d'expiration), il n'y a pas de confusion possible entre les deux pages. Toutefois, la première page ne sera plus dans le cache vu qu'elle a été écrasée par la seconde page.

Citation de l'auteur du plugin Mémoïzation :
« **Le problème de l'approche filecache** c'est le scénario où deux caches très utilisés et relativement coûteux à calculer se "marchent sur les pieds", s'annulant l'un l'autre à chaque fois qu'ils sont sollicités. »

Pour avoir une idée du comportement de la méthode « filecache » du plugin mémoization lors du passage d'un robot d'indexation ou d'un aspirateur de site, j'ai effectué l'expérimentation suivante :

- Sur un serveur de test, sur un site clone d'un site réel avec 10 000 articles.
- Utilisation d'un aspirateur de site paramétré pour éviter les images et pièces jointes.
- Aspiration lente du site (pendant 8 heures).
- Repérage du nom du fichier (F7/80) en cache de second niveau généré par un squelette gourmand, et surveillance à intervalle régulier de la date de ce fichier.

Résultat intermédiaire à 40% du nombre total de pages aspirées :

- A 40% de l'aspiration, il y avait déjà 50 317 fichiers dans le cache (224 Mo).

Résultats à la fin de l'aspiration :

- 64 488 fichiers dans le cache (sans « filecache », ce nombre serait nettement supérieur).
- 303 Mo dans le cache.
- 47% des fichiers dans le cache datent de moins d'une heure et 25 % entre 1 heure et deux heures.
- Le fichier généré par un squelette gourmand (F7/80) **a été écrasé 5 fois au cours de l'aspiration** du site, ce qui semble raisonnable.

Solution	Avantages	Inconvénients
Plugin Mémoïzation : méthode « filecache »	Limite le cache à 65 536 fichiers. Les dernières pages appelées sont dans le cache.	Le passage du robot remplit inutilement une partie du cache. Sur un site avec 235 000 fichiers dans le cache (exemple réel), le passage du robot provoquera le calcul de nombreuses pages (au moins 70 % pour ce site). « Cas où deux fichiers en cache très utilisés et coûteux à calculer s'annulent l'un l'autre à chaque fois qu'ils sont sollicités. »

3.2.2 Les autres méthodes du plugin Mémoïzation

Les autres méthodes nécessitent de disposer d'un cache d'opcode (« xcache », « apc », « eaccelerator ») et que ce dernier soit paramétré pour pouvoir stocker des variables en mémoire. Ou bien elles nécessitent le module PHP additionnel memcache qui permet de stocker des variables ou des pages entières dans la mémoire vive du serveur.

Ces solutions consistent à stocker dans la mémoire vive du serveur l'équivalent du cache de SPIP (sans la notion de répertoire).

Les 16 sous-répertoires de cache de SPIP restent alors vides, ce qui évite les limitations du nombre d'inodes.

En revanche, ces méthodes nécessitent d'avoir une mémoire vive du serveur qui permette, en plus, de stocker le volume du cache de SPIP. Pour mémoire, sur un site à forte consultation avec plus de 16 000 articles le cache a atteint 10 Go.

En cas de reboot du serveur, le cache sera vide. Aussi, le serveur doit être dimensionné pour supporter le passage d'un robot d'indexation avec un calcul de la plupart des pages.

A noter que le mécanisme de SPIP 2 permettant, via une tâche de fond, de supprimer une partie des fichiers en cache, lorsque la taille du cache dépasse le quota (cf. paragraphe 3.1), ne s'applique pas pour ces méthodes (ni pour la méthode « filecache »).

Un retour d'expérience d'un internaute sur la méthode « memcache » :
« J'utilise SPIP 2.1.13 avec memoization et la méthode memcache. J'obtiens :

- 874,66 hits par seconde de moyenne sur memcache sur 24H (hier mais à peu près les mêmes stats tous les jours)
- 29,56 « misses » par seconde de moyenne sur memcache sur 24H (hier mais à peu près les mêmes stats tous les jours)

Soit un taux d'utilisation du cache de 96,7%. Et encore il y a un autre plus petit site sous SPIP sur le même serveur qui n'utilise pas Memoization.
Pour 86 000 articles avec beaucoup d'INCLURE. Je pense donc que l'efficacité du cache de SPIP, en tout cas avec le plugin Memoization, est plus que correcte.
Il y a **2 Go alloués au cache de Memcache**. »

Solution	Avantages	Inconvénients
Plugin Mémoïzation : méthodes « xcache », « memcache », « apc », « eaccelerator »	Les 16 sous-répertoires de cache de SPIP restent alors vides, ce qui évite une saturation du nombre d'inodes utilisés ainsi qu'une saturation de l'espace disque. Les accès en mémoire sont généralement plus rapides que les accès disques.	Le serveur doit avoir une mémoire vive qui permette, en plus, de stocker le volume du cache du site SPIP (qui peut atteindre 10 Go pour un site). Le coût du Go de RAM est très supérieur au coût de Go de disque. En cas de reboot du serveur, le cache sera vide. Aussi, le serveur doit être dimensionné pour supporter le passage du robot avec un calcul de la plupart des pages.

3.3 Pour mémoire solution indiquant au robot de ne pas indexer certaines pages

Pour mémoire, lorsque l'indexation d'une page n'est pas nécessaire, il convient de l'indiquer au robot d'indexation ajoutant entre les balises <head></head> de la page :
<meta name="robots" content="noindex, nofollow" />

Lorsque l'indexation d'une page est nécessaire, mais pas celle des liens qu'elle contient, il convient de l'indiquer au robot d'indexation ajoutant entre les balises <head></head> de la page :
<meta name="robots" content="index, nofollow" />

Lorsque l'indexation d'un lien n'est pas nécessaire, il convient de l'indiquer au robot d'indexation ajoutant dans le lien : <a rel="nofollow" href="…"> ou bien <a rel="noindex,nofollow" href="…">. SPIP insère d'ailleurs automatiquement cette indication lorsqu'on utilise la balise de pagination. A noter que cette indication ne semble pas prise en compte par le robot d'indexation de Yahoo.

Par exemple, dans le plugin « cisquel : Squelettes de base avec 3 colonnes », plus de 40 squelettes contiennent depuis longtemps l'une de ces d'indications.

Solution	Avantages	Inconvénients
Pour mémoire, indiquer au robot de ne pas indexer certaines pages	Evite que le robot indexe inutilement certaines pages et les mette en cache.	Cette solution se limite aux pages (ou aux liens) dont l'indexation n'est pas nécessaire.

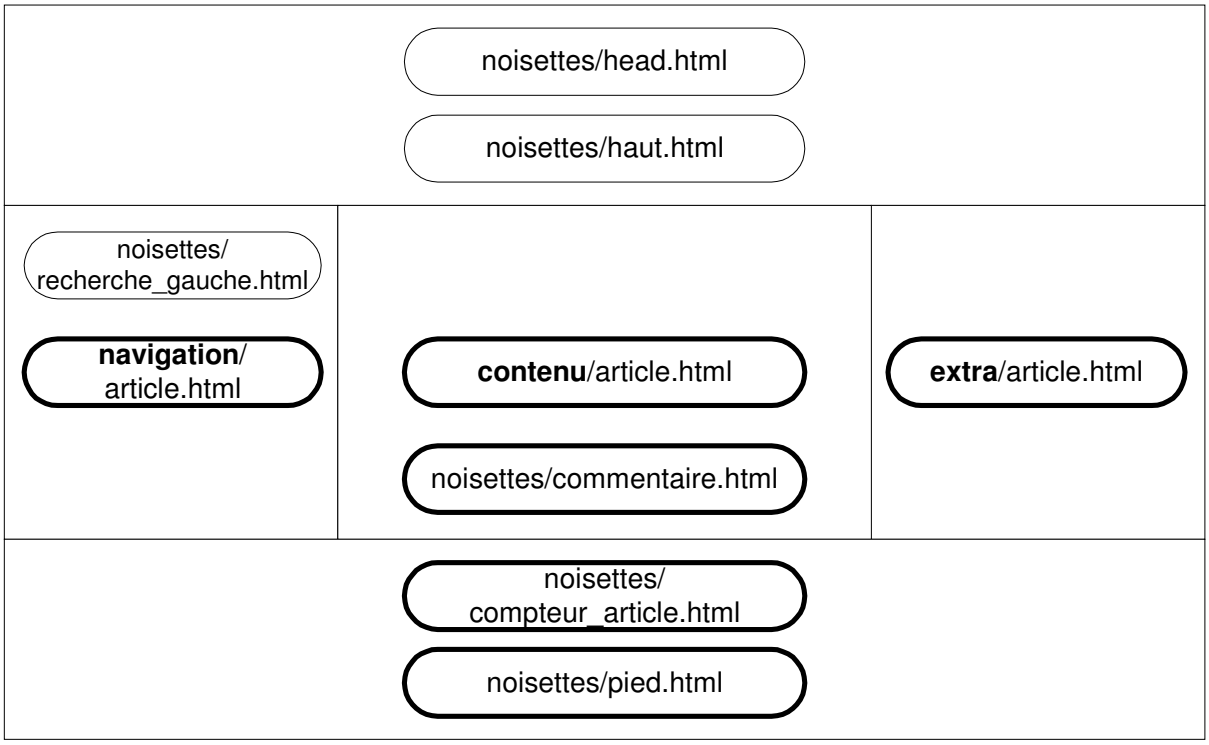
3.4 Solution consistant à utiliser la balise #INCLURE

3.4.1 Les inclusions de squelettes

Par exemple, le squelette d'article, du plugin « cisquel : Squelettes de base avec 3 colonnes », comprend des inclusions, dont certaines varient en fonction de l'article :

- navigation (colonne de gauche),
- contenu (colonne centrale)
- commentaires (colonne centrale)
- colonne de droite,
- compteur article (si le site utilise XITI)
- pied de page (s'il comprend le lien d'authentification).

Plugin des squelettes de base : page d'un article (article.html)



Cette approche permet une **meilleure factorisation**. Par exemple, la plupart des formes d'article diffèrent uniquement par le contenu de leur colonne centrale. Aussi, afin de factoriser l'ensemble de la page hors colonne centrale, le contenu de la colonne centrale (hors commentaires) fait l'objet d'une inclusion.

Comme le contenu de la colonne centrale varie d'un article à l'autre, cette inclusion génère un fichier par article dans le cache de second niveau de SPIP. Idem pour chaque inclusion entourée d'un trait gras dans le schéma ci-dessus. Aussi, **pour un article on peut avoir jusqu'à 6 fichiers d'inclusion dans le cache de second niveau de SPIP** (plus un fichier de cache pour le squelette principal).

Ces inclusions restent indispensables, pour des raisons de factorisation, de coût de maintenance, etc. Il ne s'agit pas de les remettre en cause. En revanche, **il convient d'optimiser leur incidence sur le cache** de second niveau de SPIP en exploitant une nouveauté introduite par SPIP 1.9.

3.4.2 Le mode d'inclusion classique

Le mode d'inclusion classique <INCLURE{fond=squeletteB}> est apparu en premier.

Le squelette B possède son propre cache de second niveau et le squelette principal possède aussi son propre cache.

Lorsque l'on demande la page, le cache du squelette B et celui du squelette principal sont lus et la combinaison des deux est envoyée au navigateur.

<INCLURE> permet d'avoir un squelette B n'ayant pas la même durée de cache que le squelette principal. C'est particulièrement utile pour le menu déroulant du site, car cela évite de le calculer trop souvent.

3.4.3 La balise #INCLURE

Le second mode d'inclusion #INCLURE{fond=B} a été introduit par SPIP 1.9.1 (la balise #INCLURE).

Le cache de second niveau du squelette B est placé dans le cache du squelette principal.

La balise #INCLURE ne permet pas de choisir la durée de cache du squelette B, car il sera calculé en même temps que le squelette principal. C'est donc à éviter pour le menu déroulant du site.

Contrairement à ce qui est indiqué dans la documentation de SPIP, si l'on utilise la balise #INCLURE pour inclure la colonne de droite, il y a des fichiers générés dans le cache de second niveau pour la colonne de droite. C'est effectivement un bug de la documentation de SPIP :
Cf. http://archives.rezo.net/archives/spip-dev.mbox/YBPS43DRMXK3CW6VMVVLNDRZW355PGRF/

Aussi, la balise #INCLURE ne semble pas pouvoir réduire le nombre de fichiers dans le cache de second niveau de SPIP.

3.4.4 Utilisation combinée de la balise #INCLURE et de #CACHE{0}

J'ai remarqué que si le squelette B commence par la balise #CACHE{0} et que le squelette principal utilise la balise #INCLURE pour le squelette B, alors il n'y a pas de fichier généré dans le cache de second niveau pour le squelette B.

A noter que le squelette principal n'est pas recalculé à chaque appel. Ceci est normal puisque le cache de second niveau du squelette B est placé dans le cache du squelette principal.

Il convient d'appliquer avec discernement cette approche. Par exemple, pour le squelette d'article, du plugin « cisquel : Squelettes de base avec 3 colonnes », il convient d'utiliser la balise #INCLURE uniquement pour les 4 inclusions suivantes :

- navigation/article
- noisettes/commentaire
- extra/article
- noisettes/compteur_article

Il convient de **ne pas utiliser la balise #INCLURE pour « contenu/article »** en raison des cas où le squelette inclus doit avoir une durée de cache différente du squelette principal et des cas où il contient un formulaire #FORMULAIRE_…

Remarques importantes d'un développeur du noyau de SPIP :

- L'utilisation de la balise #INCLURE suppose qu'il n'y ait pas de #FORMULAIRE_xxx ou de balise dynamique dans le fichier inclus.
- Si le fichier inclus est appelé par un autre squelette via une inclusion classique, il sera recalculé à chaque fois en raison de l'utilisation de #CACHE{0}.

3.4.5 Cas particulier de la balise #SESSION (dans le pied de page)

Concernant le pied de page, des expérimentations ont montré que, **s'il comprend la balise #SESSION** (pour le lien authentification / déconnexion), **il est préférable d'utiliser l'inclusion classique**.

Par ailleurs, la balise #SESSION dans le pied de page génère deux fichiers par article dans le cache de second niveau de SPIP (le second se distingue par un caractère souligné à la fin du nom) :

- noi--8-2012_06_06%2010%3-b04dd38f
- noi--8-2012_06_06%2010%3-b04dd38f_

Dans cet exemple, le premier fichier contient une seule ligne :
a:2:{s:11:"invalideurs";a:1:{s:7:"session";s:0:"";}s:12:"lastmodified";i:1338969989;}

Si on remplace l'utilisation de la balise #SESSION :

```
[(#SESSION{id_auteur}|?{' '})| <a href="#URL_LOGOUT" rel="noindex,nofollow"
title="<:cisquel:eq_deconnexion:>"><:cisquel:eq_deconnexion:></a>
[(#SESSION{login})] ]
[(#SESSION{id_auteur}|?{'',' '})| <a
href="[(#URL_PAGE{login}|parametre_url{url,#SELF})]" rel="noindex,nofollow"
title="<:cisquel:eq_authentification:>"><:cisquel:eq_authentification:></a>]
```

par l'utilisation de PHP (ce qui est à éviter dans les squelettes) :

```
<?php if ($GLOBALS["visiteur_session"]["id_auteur"]) { ?>
<a href="#URL_LOGOUT" rel="noindex,nofollow"
title="<:cisquel:eq_deconnexion:>"><:cisquel:eq_deconnexion:></a>
<?php echo interdire_scripts(entites_html($GLOBALS["visiteur_session"]["login"]));
} else { ?>
<a href="[(#URL_PAGE{login}|parametre_url{url,#SELF})]" rel="noindex,nofollow"
title="<:cisquel:eq_authentification:>"><:cisquel:eq_authentification:></a>
<?php } ?>
```

alors le pied de page génère un seul fichier par article dans le cache de second niveau de SPIP.

Enfin, **l'utilisation de PHP** au lieu de la balise #SESSION et l'utilisation de #CACHE{0}, pour le pied de page, permet à ce dernier de ne pas générer de fichier par article dans le cache de second niveau de SPIP.

Remarques :

a) Dans cet exemple, il est nécessaire d'utiliser l'inclusion classique pour le pied de page. En effet, la balise #INCLURE ne permet pas de choisir la durée de cache du squelette inclus, car il sera calculé en même temps que le squelette principal, aussi le code PHP ne serait exécuté que lors du calcul du squelette principal.

b) Si le pied de page contient des boucles, en plus du lien authentification / déconnexion, il est intéressant de le décomposer en deux squelettes, pour éviter de calculer les boucles à chaque consultation. Par exemple :
- pied.html : avec les boucles et sans #CACHE{0}
- pied_connexion.html : avec le lien authentification / déconnexion et avec #CACHE{0}

3.4.6 Mesure des résultats

Avec une utilisation combinée de la balise #INCLURE et de #CACHE{0}, ainsi qu'en utilisant PHP au lieu de la balise #SESSION, la consultation de 1000 articles sur un site de test génère 2003 fichiers (2xN + 3) dans le cache de second niveau, au lieu de 6003 fichiers auparavant (6xN + 3).

Le nombre de fichiers dans le cache de second niveau est divisé par 3

Les 2003 fichiers représentent au total 22,8 Mo, contre 34,6 Mo pour les 6003 fichiers auparavant.

La taille du cache diminue d'environ 35 %

Les temps de traitement d'un article n'augmentent pas, qu'il soit dans le cache de second niveau, qu'il soit calculé ou qu'il soit recalculé. Par ailleurs, un test de charge sur un article dans le cache de second niveau a confirmé que les performances sont préservées.

Les temps de traitements n'augmentent pas

Solution	Avantages	Inconvénients
Utiliser la balise #INCLURE (et utiliser PHP au lieu de la balise #SESSION dans le pied de page)	Le nombre de fichiers dans le cache de second niveau est divisé par 3. La taille du cache diminue d'environ 35 %. Les temps de traitements n'augmentent pas.	Cela suppose qu'il n'y ait pas de #FORMULAIRE_xxx ou de balise dynamique dans le fichier inclus. Ainsi, dans l'exemple de squelette article, la balise #INCLURE n'est pas utilisée pour « contenu/article ».

3.5 Solution consistant à ne pas mettre en cache les pages lors du passage du robot

ATTENTION : Cette solution nécessite que le serveur soit dimensionné pour supporter le passage d'un robot d'indexation avec un calcul de la plupart des pages. Je décline toute responsabilité en cas de surcharge du serveur.

Cette solution consiste à ne pas mettre les pages dans le cache de second niveau (sauf pour certains squelettes communs aux pages ou gourmands) lors du passage d'un robot d'indexation (avec une liste de robots paramétrable). Ceci évite de remplir inutilement le cache, puis de le vider (cf. 3.1), puis de recommencer lors du prochain passage du robot.

Mettre dans un fichier d'options les lignes suivantes **en adaptant** les paramètres :

```
// ne pas mettre en cache les pages lors du passage du robot
define('_BOT_PAS_METTRE_EN_CACHE','oui');

// partie de user_agent des robots
define('_BOT_LISTE',',bot|mnogosearch|slurp|crawler|spider|webvac|yandex,i');

// liste des squelettes communs aux pages ou gourmands à mettre quand même en cache
define('_BOT_SKEL_A_CACHER','noisettes/haut,noisettes/head,noisettes/pied,noisettes
/recherche_gauche,archive_article,archive_article_rubrique,noisettes/archive_articl
e_annee');
```

Mettre dans un plugin, le fichier public/assembler.php avec le contenu suivant :

```
<?php
if (!defined('_ECRIRE_INC_VERSION')) return;
include_spip('ecrire/public/assembler');

function public_produire_page($fond, $contexte, $use_cache, $chemin_cache,
$contexte_cache, &$page, &$lastinclude, $connect=''){
	$parametrer = charger_fonction('parametrer', 'public');
	$page = $parametrer($fond, $contexte, $chemin_cache, $connect);

	// si robot d'indexation, ne pas stocker en cache sauf les exceptions
	$ok = true;
	if (defined('_BOT_PAS_METTRE_EN_CACHE') AND _BOT_PAS_METTRE_EN_CACHE=='oui'){
		if (_IS_BOT OR defined('_BOT_LISTE')){
			if (_IS_BOT OR (isset($_SERVER['HTTP_USER_AGENT']) AND
preg_match(_BOT_LISTE,$_SERVER['HTTP_USER_AGENT']))) {
				if (defined('_BOT_SKEL_A_CACHER')) {
					$tableau_fonds = explode(',', _BOT_SKEL_A_CACHER);
					if (!in_array($fond,$tableau_fonds))
						$ok = false;
				} else {
					$ok = false;
				}
			}
		}
	}

	if ($ok) {
	// et on l'enregistre sur le disque
	if ($chemin_cache
	AND $use_cache>-1
	AND is_array($page)
	AND count($page)
	AND $page['entetes']['X-Spip-Cache'] > 0){
		$cacher = charger_fonction('cacher', 'public');
		$lastinclude = time();
		$cacher($contexte_cache,$use_cache,$chemin_cache,$page,$lastinclude);
	}
	}
	return $page;
}
?>
```

Solution	Avantages	Inconvénients
Ne pas mettre en cache les pages lors du passage du robot (sauf certains squelettes)	Au passage du robot, évite de remplir inutilement le cache (puis de le vider ultérieurement). Permet de mettre quand même en cache les squelettes communs aux pages ou gourmands.	Le serveur doit être dimensionné pour supporter le passage du robot avec un calcul de la plupart des pages.

4. Résumé des avantages et inconvénients des différentes solutions

Solution	Avantages	Inconvénients
Solution native de SPIP de limitation de la taille du cache	Limite la taille du cache de SPIP à un quota que l'on peut fixer.	« La taille réelle peut être supérieure au quota ». « Si le nombre de fichiers est trop élevé, le mécanisme n'arrive même plus à vider le cache, si le serveur est trop lent ». Si le quota est trop faible, le passage du robot provoquera le calcul de nombreuses pages.
Plugin Mémoïzation : méthode « filecache »	Limite le cache à 65 536 fichiers. Les dernières pages appelées sont dans le cache.	Le passage du robot remplit inutilement une partie du cache. Sur un site avec 235 000 fichiers dans le cache (exemple réel), le passage du robot provoquera le calcul de nombreuses pages (au moins 70 % pour ce site). « Cas où deux fichiers en cache très utilisés et coûteux à calculer s'annulent l'un l'autre à chaque fois qu'ils sont sollicités. »
Plugin Mémoïzation : méthodes « xcache », « memcache », « apc », « eaccelerator »	Les 16 sous-répertoires de cache de SPIP restent alors vides, ce qui évite une saturation du nombre d'inodes utilisés ainsi qu'une saturation de l'espace disque. Les accès en mémoire sont généralement plus rapides que les accès disques.	Le serveur doit avoir une mémoire vive qui permette, en plus, de stocker le volume du cache du site SPIP (qui peut atteindre 10 Go pour un site). Le coût du Go de RAM est très supérieur au coût de Go de disque. En cas de reboot du serveur, le cache sera vide. Aussi, le serveur doit être dimensionné pour supporter le passage du robot avec un calcul de la plupart des pages.
Pour mémoire, indiquer au robot de ne pas indexer certaines pages	Evite que le robot indexe inutilement certaines pages et les mette en cache.	Cette solution se limite aux pages (ou aux liens) dont l'indexation n'est pas nécessaire.
Utiliser la balise #INCLURE (et utiliser PHP au lieu de la balise #SESSION dans le pied de page)	Le nombre de fichiers dans le cache de second niveau est divisé par 3. La taille du cache diminue d'environ 35 %. Les temps de traitements n'augmentent pas.	Cela suppose qu'il n'y ait pas de #FORMULAIRE_xxx ou de balise dynamique dans le fichier inclus. Ainsi, dans l'exemple de squelette article, la balise #INCLURE n'est pas utilisée pour « contenu/article ».
Ne pas mettre en cache les pages lors du passage du robot (sauf certains squelettes)	Au passage du robot, évite de remplir inutilement le cache (puis de le vider ultérieurement). Permet de mettre quand même en cache les squelettes communs aux pages ou gourmands.	Le serveur doit être dimensionné pour supporter le passage du robot avec un calcul de la plupart des pages.

Certaines solutions sont complémentaires.

Exemples de combinaisons de solutions	**Exemple 1**	**Exemple 2** (a)	**Exemple 3** (b)	**Exemple 4** (a)	**Exemple 5** (a)(c)	**Exemple 6** (a)(c)
Solution native de SPIP de limitation de la taille du cache	X	X	(d)	(d)	(d)	(d)
Plugin Mémoïzation : méthode « filecache »			X	X		
Plugin Mémoïzation : méthodes « xcache », « memcache », « apc », « eaccelerator »					X	X
Pour mémoire, indiquer au robot de ne pas indexer certaines pages	X	X	X	X	X	X
Utiliser la balise #INCLURE (et utiliser PHP au lieu de la balise #SESSION dans le pied de page)	X	X	X	X	X	X
Ne pas mettre en cache les pages lors du passage du robot (sauf certains squelettes)		X		X		X

(a) le serveur doit être dimensionné pour supporter le passage du robot avec un calcul de la plupart des pages.
(b) le serveur doit être dimensionné pour supporter le passage du robot avec un calcul de nombreuses pages.
(c) le serveur doit avoir une mémoire vive qui permette, en plus, de stocker le volume du cache du ou des sites SPIP.
(d) ne s'applique pas lorsqu'on utilise le plugin Mémoïzation.

L'exemple 2 offre une véritable prévention vis-à-vis des robots d'indexation.

L'exemple 4, qui reprend l'exemple 2 en lui ajoutant une solution additionnelle, peut être intéressant dans le cas d'un site à très forte volumétrie.

L'exemple 6, qui reprend l'exemple 2 en lui ajoutant une solution additionnelle, peut être intéressant dans le cas d'un site à très fort trafic. Dans ce cas, le serveur doit avoir une mémoire vive qui permette, en plus, de stocker le volume du cache du site SPIP. Toutefois, il convient de comparer les performances de cet exemple avec celui d'un reverse proxy cache (par exemple « Varnish »).