Optimiser le cœur de votre site face aux bots IA

Bernard Sfez - 2025-11-26 15:16

Tiki Wiki CMS est au cœur de plusieurs sites qui subissent les requêtes de crawlers d'IA, agressifs et voraces. Ces derniers sont capables de saturer les serveurs au point de rendre les applications lentes ou carrément inaccessibles comme nous l'avons montré dans un précédent article.

La réponse ne peut pas se limiter à ce qui se passe « à la porte du serveur » (WAF, rate limiting, blocage d'IP, etc.). Il est tout aussi essentiel de muscler Tiki Wiki lui-même, en agissant sur :

- Le contenu que le site expose réellement aux visiteurs et aux robots
- La façon dont les requêtes (queries) sont construites, filtrées et enchaînées
- La détection et la maîtrise des consommations anormales de mémoire ou de CPU
- Les données qui sont interrogées et celles qui sont effectivement renvoyées
- L'utilisation des outils d'optimisation et de performance propres à Tiki.

L'objectif de cet article est de présenter des mesures concrètes dans Tiki Wiki CMS pour réduire la surface d'attaque, maîtriser la consommation de ressources et garantir la disponibilité de votre site, même sous la pression de bots d'IA ou de crawlers malveillants, tout en améliorant durablement ses performances.

Cartographier et neutraliser les brèches

Les bots d'IA vont partout et n'explorent plus uniquement vos pages publiques de navigation principale. Ils suivent tous les liens, y compris ceux oubliés, et déclenchent toutes sortes de requêtes complexes (recherches, filtres avancés, exports, vues « historiques » construites il y a des années, etc.). Il est rare de ne pas trouver quelques leftovers – pages de test, pages incomplètes ou anciennes fonctionnalités – sur les sites dont nous prenons la responsabilité. Il est impératif de reprendre le contrôle de ce qui est réellement exposé.

Aller fouiller l'information directement dans les logs d'accès de votre serveur Apache, Nginx ou autre n'est pas la solution idéale au quotidien. Si votre hébergement propose un panneau de contrôle (control panel), vous disposez souvent d'outils de type AWStats. Dans tous les cas,

activez ou mettez en place un outil de statistiques pour visualiser les pages effectivement visitées (et donc visitables) et prenez l'habitude de les consulter régulièrement : idéalement tous les jours au début, puis au moins une fois par semaine, voire une fois par mois lorsque la situation est stabilisée.

Dans nos packages de maintenance, nous nous appuyons sur les statistiques de fréquentation de Matomo, que nous installons sur des serveurs privés (les nôtres ou ceux de nos clients). Matomo permet d'identifier rapidement :

- Les URL les plus sollicitées par les bots,
- Les paramètres de requêtes qui déclenchent des traitements lourds,
- Les pics d'accès anormaux sur des pages anciennes ou supposées confidentielles.

Cette solution d'analytics open source permet de conserver 100 % des données sur vos propres serveurs, ce qui est essentiel pour la souveraineté et la conformité au RGPD.

Revoir les permissions (et les catégories) pour ce qui doit rester privé

Une fois cette cartographie en place, il est impératif de :

- Réviser finement les permissions Tiki (Trackers, fiches d'administration, vues d'export, pages de rapport, etc.) afin d'éviter que des fonctionnalités très gourmandes en ressources soient accessibles en anonyme ou à des groupes trop larges.
- Nettoyer la navigation (menus, liens internes, sitemaps, pages d'index) pour retirer les pages techniques, historiques ou de test qui n'ont plus de raison d'être publiques mais restent pourtant atteignables par les bots.
- Utiliser les catégories et leur système de permissions pour isoler les zones sensibles (administration, reporting, exports, données internes) et contrôler précisément qui peut y accéder.
- Limiter l'indexation (robots, liens, sitemaps) des sections qui ne doivent pas être explorées massivement, même si elles restent accessibles à certains profils authentifiés.

Sur un Tiki moderne, l'enjeu n'est pas seulement de décider « qui peut voir », mais surtout de maîtriser « quelles requêtes complexes un bot peut déclencher sans contrôle » sur vos Trackers, listes et recherches avancées.

Être alerté en cas de surconsommation avec Zabbix

Zabbix est une solution de supervision open source de niveau entreprise. Elle offre une visibilité détaillée sur la charge CPU, la mémoire, les I/O disque, les connexions réseau et l'état de vos services (PHP-FPM, MariaDB, Elasticsearch/Manticore, etc.), avec la possibilité de définir des seuils et des alertes en temps réel.

En corrélant Matomo (ce qui est visité) et Zabbix (ce qui souffre), on voit immédiatement :

- Quelles URL déclenchent des pics CPU sur MariaDB.
- Quels scripts PHP épuisent les workers PHP-FPM.
- Quels créneaux horaires sont spécifiquement ciblés par des crawlers.

Cette visibilité croisée permet de repérer très tôt les comportements anormaux et d'intervenir avant que la surcharge ne se transforme en indisponibilité.

Optimiser la pile technique Tiki + serveur pour éviter l'effet boule de neige

Même un site bien conçu peut tomber si la couche serveur est mal dimensionnée ou mal optimisée. Un premier problème de sur-utilisation de la mémoire peut rapidement déclencher une chaîne de défaillances.

Le scénario classique : CPU ou mémoire saturée jusqu'à l'indisponibilité

Sous charge extrême (bots d'IA, vagues de crawlers), on voit souvent le scénario suivant :

1. MariaDB consomme de plus en plus de mémoire à cause de requêtes lourdes ou répétées.
2. Le système commence à swapper ou le noyau tue des processus (OOM killer).
3. PHP-FPM se met à redémarrer, perd ses workers ou refuse de nouvelles connexions.
4. Les files de messages et de tâches internes se remplissent, entraînant un rallongement des temps de réponse, puis des timeouts.
5. Au final, le site devient indisponible pour tout le monde.

On a vu récemment une illustration spectaculaire de ce type de cascade avec la panne AWS d'octobre 2025 : un problème de DNS autour de DynamoDB a déclenché une série d'incidents internes et une accumulation de messages en file d'attente, impactant plus d'une centaine de services AWS et laissant de nombreux sites indisponibles pendant plusieurs heures.
L'enseignement à retenir : un petit problème local (DNS, mémoire, file d'attente) peut mettre à genoux une architecture entière si les couches au-dessus ne sont pas conçues pour limiter leur consommation.

Optimiser les outils et bibliothèques autour de Tiki

Pour un Tiki robuste face aux bots :
PHP-FPM

- Adapter le nombre de workers (pm.max_children, pm.max_requests) à la RAM réelle.
- Fixer des limites de temps (max_execution_time) et de mémoire (memory_limit) raisonnables pour éviter qu'une seule requête ne monopolise les ressources.

MariaDB

- Surveiller les requêtes lentes (slow query log).
- Ajuster les paramètres de buffers (InnoDB buffer pool) à la taille de vos données.
- Nettoyer ou optimiser les index utilisés par vos Trackers les plus sollicités.

Moteur de recherche (MySQL, Elasticsearch, Manticore)
Tiki repose sur un moteur de recherche unifié qui peut s'appuyer sur différents backends (MySQL/MariaDB, Elasticsearch ou Manticore).

- Sur des installations volumineuses, Elasticsearch ou Manticore apportent un vrai gain pour les recherches complexes et les très gros volumes de données.
- Mais un mauvais paramétrage de ces moteurs peut aussi amplifier la consommation mémoire.
- Installer les outils de monitoring nécessaire à l'ajustement des paramètres en visant l'optimisation.

Outils de build
Avec Tiki 29, les exigences de build et d'installation ont augmenté (par exemple la dépendance à Node.js 20 pour la compilation des assets), ce qui peut entraîner des erreurs « out of memory » sur des serveurs sous-dimensionnés pendant l'installation de Tiki Wiki ou les mises à jour. Ce n'est pas directement lié à la charge des bots, mais cela illustre à quel point la marge mémoire doit être confortable sur un serveur Tiki moderne.

Refondre les pages Tiki les plus lourdes

Les bots d'IA ne se contentent pas de charger la page d'accueil. Ils enchaînent les requêtes, les filtres et les combinaisons d'URL, parfois des milliers de fois, sur des pages construites il y a longtemps.

Si la famille des plugins Tracker a permis à Tiki de briller pendant des années, elle a été progressivement concurrencée, puis en grande partie remplacée, par les plugins List, CustomSearch et ListExecute, introduits il y a plus d'une dizaine d'années et bien mieux adaptés aux usages actuels.

Chez OpenSource Solutions, nous n'utilisons plus cette ancienne génération de plugins sur les nouveaux développements depuis longtemps. Cependant, il n'est pas rare de les retrouver sur des Tiki en production depuis de nombreuses années. Il faut donc rechercher en priorité les pages :

- qui utilisent le plugin Tracker ou TrackerList pour construire des listes complexes, avec plusieurs jointures et filtres ;
- qui produisent des affichages dimensionnés pour un débit humain (quelques dizaines de consultations par heure), mais qui deviennent catastrophiques quand un bot les appelle en rafale.

Chaque page, prise isolément, peut sembler acceptable en termes de performance. Mais lorsqu'un crawler d'IA simule des centaines de requêtes en chaîne, la charge sur MariaDB et PHP-FPM explose.

Remplacer les Trackers lourds par List, CustomSearch et ListExecute

Une approche moderne et beaucoup plus robuste consiste à refondre ces pages avec des plugins qui s'appuient sur le Unified Index de Tiki.

Plugin List

- Conçu pour exécuter des requêtes complexes sur de gros volumes de données, en s'appuyant sur le moteur d'indexation (MySQL, Elasticsearch, Manticore).
- Permet une adaptation fine et optimisée des écrans et des interfaces (mise en page, colonnes, formats d'affichage, etc.).

Plugin CustomSearch

- Permet de construire des interfaces de recherche avancées et ergonomiques, tout en maîtrisant précisément les filtres et les champs exposés.
- Offrant les mêmes possibilités de personnalisation des écrans et des interfaces, il devient la brique idéale pour créer des formulaires de recherche performants et adaptés à vos usages métiers.

Plugin ListExecute

- Permet d'exécuter des actions à partir d'une liste (workflow interne, commandes programmées, opérations sur un ensemble d'objets, etc.), tout en restant dans le cadre du Unified Index et d'un comportement maîtrisé.
- Conçu pour une interopérabilité étroite avec le planificateur Tiki (Tiki Scheduler) afin d'automatiser certaines opérations.
- Bénéficie lui aussi d'une adaptation optimisée des écrans et interfaces pour proposer des actions claires et sécurisées aux utilisateurs.

Ce refactoring présente plusieurs avantages :

- Déléguer les recherches au moteur d'index (spécialisé et optimisé pour cela).
- Réduire drastiquement le nombre de requêtes SQL lourdes exécutées par MariaDB.
- Mieux contrôler les champs réellement chargés et affichés.
- Mieux résister à des charges anormales générées par des bots.

Le résultat concret, mesuré via Zabbix et les outils du serveur, est une baisse significative de l'utilisation CPU et mémoire de MariaDB et des workers PHP-FPM sur ces pages, même sous forte pression de crawlers.

Limiter les champs récupérés pour réduire la mémoire (à partir de Tiki29)

Les installations Tiki avancées (intranet, portails métiers, bases de connaissances) peuvent atteindre des index contenant des milliers de champs de Trackers. Dans ce contexte, chaque requête qui « récupère tout » devient coûteuse, surtout si l'on fait des calculs ou des sous-totaux sur un grand nombre de résultats.

Nouveau paramètre "fields" dans les listes (select context)

À partir de la branche Tiki 29 , un commit d'amélioration ajoute la possibilité de sélectionner explicitement les champs à récupérer dans le contexte d'un plugin List / CustomSearch, afin de réduire la consommation mémoire.

Quelques points importants :

- Seuls les champs listés dans `fields` seront effectivement récupérés depuis l'index de recherche.
- Cette capacité s'appuie sur la couche d'index unifiée de Tiki, commune aux trois moteurs supportés (MySQL, Elasticsearch, Manticore).(Tiki4)
- Les champs techniques `object_type` et `object_id` restent toujours disponibles, même s'ils ne sont pas mentionnés.

Cette optimisation offre de nombreux avantages:

- On ne récupère que trois champs métier (titre, statut, propriétaire),
- Tous les autres champs du tracker restent dans l'index mais ne sont pas chargés en mémoire pour cette liste,
- Sur un index avec des milliers de champs et des dizaines de milliers d'items, cet allègement peut faire une différence significative en termes de RAM consommée et de temps de traitement pour les calculs et sous-totaux.

La même logique s'applique dans les contextes où CustomSearch s'appuie sur le Unified Index pour exécuter des requêtes complexes.
Si vous utilisez déjà les plugins List ou CustomSearch mais sans tirer parti de ce paramètre, OpenSource Solutions peut analyser vos pages Tiki existantes, identifier les listes les plus coûteuses, implémenter correctement fields et optimiser vos écrans. Nous pouvons également mettre à jour votre Tiki vers une version adaptée afin que vous puissiez bénéficier de ces nouveautés dans des conditions stables et sécurisées. Vous pouvez nous confier cette refonte ciblée pour réduire durablement la consommation de ressources de votre Tiki Wiki CMS.

Un résumé d'une démarche pro-active pour votre Tiki Wiki

Face aux crawlers d'IA et aux bots de plus en plus agressifs, la défense ne peut pas se limiter au

firewall ou au serveur web. Il est crucial d’aligner :

1. La couche de supervision

- Matomo pour voir *qui* consomme *quelles pages* et *comment*,
- Zabbix pour voir *comment le serveur encaisse* (CPU, mémoire, disque, réseau).

2. La configuration serveur
   - PHP-FPM dimensionné correctement,
   - MariaDB optimisé pour vos usages réels,
   - Moteur de recherche adapté à votre volume et type de données (MySQL, Elasticsearch, Manticore).

3. L’architecture Tiki
   - Permissions revues pour limiter ce que les bots peuvent déclencher,
   - Refactoring des pages les plus lourdes (Trackers → List / CustomSearch / ListExecute),
   - Usage systématique des nouvelles capacités comme `fields` pour réduire la mémoire consommée par le Unified Index.

Ainsi, votre Tiki ne se contente plus de « survivre » aux bots d’IA : il reste rapide, disponible et maîtrisé, même quand les crawlers se déchaînent.