How to Keep Your Site from Sinking Under AI Bots

How to Keep Your Site from Sinking Under AI Bots

Bernard Sfez - 2025-11-27 15:46



Tiki Wiki CMS is used on several sites that are currently under heavy fire from aggressive, resource-hungry AI crawlers. These bots can overwhelm servers to the point where applications become painfully slow or even completely unavailable, as we discussed in a previous article.

The answer cannot be limited to what happens “at the server’s doorstep” (WAF, rate limiting, IP blocking, etc.). It is just as essential to strengthen Tiki Wiki itself by acting on:

- The content the site actually exposes to visitors and bots
- The way queries are built, filtered, and chained
- The detection and control of abnormal memory or CPU consumption
- The data that is queried versus what is actually returned
- The use of Tiki’s own optimisation and performance tools

The goal of this article is to present concrete measures within Tiki Wiki CMS to reduce the attack surface, control resource consumption, and ensure your site stays available even under AI: crawlers hungry for data that overload your servers and leave your sites slow or inaccessible|pressure from AI bots or malicious crawlers], while also improving its performance over the long term.

Mapping and neutralizing open doors

AI bots now go everywhere and no longer explore only your main public navigation pages. They follow every link, including forgotten ones, and trigger all kinds of complex requests (searches, advanced filters, exports, “historical” views built years ago, etc.). It is rare not to find a few leftovers – test pages, incomplete pages, or legacy features – on the sites we take responsibility for. It is essential to regain control over what is actually exposed.

Digging directly into the access logs of your Apache, Nginx or other web server is not an ideal day-to-day solution. If your hosting provides a control panel, you often have tools such as

AWStats available. In all cases, enable or set up a statistics tool to visualize which pages are actually visited (and therefore visitable), and get into the habit of consulting these stats regularly: ideally every day at the beginning, then at least once a week, and eventually once a month once the situation has stabilized.

In our maintenance packages, we rely on the traffic statistics of Matomo, which we install on private servers (ours or our clients'). Matomo makes it easy to identify:

- The URLs most targeted by bots,
- The query parameters that trigger heavy processing,
- Abnormal access spikes on old or supposedly confidential pages.

This open source analytics solution allows you to keep 100% of the data on your own servers, which is essential for data sovereignty and GDPR compliance.

Reviewing permissions (and categories) for what must remain private

Once this mapping is in place, it is imperative to:

- Carefully review Tiki permissions (Trackers, administration records, export views, report pages, etc.) to avoid having very resource-hungry features accessible anonymously or to overly broad groups.
- Clean up navigation (menus, internal links, sitemaps, index pages) to remove technical, historical, or test pages that no longer have any reason to be public but are still reachable by bots.
- Use categories and their permission system to isolate sensitive areas (administration, reporting, exports, internal data) and precisely control who can access them.
- Limit indexing (robots, links, sitemaps) of sections that should not be massively crawled, even if they remain accessible to certain authenticated profiles.

On a modern Tiki, the issue is not only deciding "who can see what," but above all controlling "which complex queries a bot can trigger without restriction" on your Trackers, lists, and advanced searches.

Getting alerts on overconsumption with Zabbix

Zabbix is an open source, enterprise-grade monitoring solution. It provides detailed visibility into CPU load, memory, disk I/O, network connections, and the state of your services (PHP-FPM, MariaDB, Elasticsearch/Manticore, etc.), with the ability to define thresholds and real-time alerts.

By correlating Matomo (what is being visited) and Zabbix (what is suffering), you can immediately see:

- Which URLs trigger CPU spikes on MariaDB.
- Which PHP scripts exhaust PHP-FPM workers.

- Which time slots are specifically targeted by crawlers.

This combined visibility lets you detect abnormal behavior very early and intervene before overload turns into outright downtime.

Optimizing the Tiki + server stack to avoid the snowball effect

Even a well-designed site can go down if the server layer is poorly sized or badly tuned. An initial memory overuse problem can quickly trigger a chain reaction of failures.

The classic scenario: CPU or memory saturation leading to downtime

Under extreme load (AI bots, waves of crawlers), the following pattern is very common:

1. MariaDB consumes more and more memory because of heavy or repeated queries.
2. The system starts to swap, or the kernel kills processes (OOM killer).
3. PHP-FPM starts restarting, loses workers, or refuses new connections.
4. Internal message and task queues fill up, causing longer response times, then timeouts.
5. In the end, the site becomes unavailable for everyone.

We recently saw a spectacular illustration of this kind of cascade with the AWS outage in October 2025: a DNS issue around DynamoDB triggered a series of internal incidents and an accumulation of queued messages, impacting more than a hundred AWS services and leaving many sites unavailable for several hours.
The key lesson: a small local issue (DNS, memory, queue) can bring an entire architecture to its knees if the layers above are not designed to limit their resource consumption.

Optimizing the tools and libraries around Tiki

To make Tiki robust against bots:

PHP-FPM

- Tune the number of workers (pm.max_children, pm.max_requests) to match the actual RAM.
- Set reasonable time (max_execution_time) and memory (memory_limit) limits to prevent a single request from monopolizing resources.

MariaDB

- Monitor slow queries (slow query log).
- Adjust buffer settings (InnoDB buffer pool) to the size of your data.

- Clean up or optimize indexes used by your most heavily used Trackers.

Search engine (MySQL, Elasticsearch, Manticore)
Tiki relies on a unified search engine that can use different backends (MySQL/MariaDB, Elasticsearch, or Manticore).

- On large installations, Elasticsearch or Manticore bring real benefits for complex searches and very large data volumes.
- But poor tuning of these engines can also amplify memory consumption.
- Install the monitoring tools needed to tune parameters with optimization in mind.

Build-in tools
With Tiki 29, build and installation requirements have increased (for example, the dependency on Node.js 20 for asset compilation), which can cause “out of memory” errors on undersized servers during Tiki Wiki installation or upgrades. This is not directly related to bot load, but it illustrates how important it is to have a comfortable memory margin on a modern Tiki server.

Reworking the heaviest Tiki pages

AI bots do not just load the home page. They chain requests, filters, and URL combinations, sometimes thousands of times, on pages that were built a long time ago.

While the Tracker plugin family allowed Tiki to shine for years, it has gradually been challenged and largely replaced by the List, CustomSearch, and ListExecute plugins, introduced more than ten years ago and much better suited to current use cases.

At OpenSource Solutions, we have not used this older generation of plugins for new development for a long time. However, it is still common to find them on Tiki instances that have been in production for many years. You should therefore prioritize looking for pages:

- that use the Tracker or TrackerList plugin to build complex lists with multiple joins and filters;
- that produce views sized for human traffic (a few dozen views per hour), but become disastrous when a bot calls them in rapid succession.

Each page, taken in isolation, may seem acceptable in performance terms. But when an AI crawler simulates hundreds of chained requests, **load on MariaDB and PHP-FPM explodes**.

Replacing heavy Trackers with List, CustomSearch, and ListExecute

A modern and much more robust approach is to redesign these pages with plugins that rely on Tiki’s Unified Index.

Plugin List

- Designed to run complex queries on large volumes of data using the indexing engine (MySQL, Elasticsearch, Manticore).
- Allows fine-tuned, optimized adaptation of screens and interfaces (layout, columns, display formats, etc.).

Plugin CustomSearch

- Allows you to build advanced, user-friendly search interfaces while precisely controlling which filters and fields are exposed.
- Offering the same possibilities for screen and interface customization, it becomes the ideal building block for creating high-performance search forms tailored to your business use cases.

Plugin ListExecute

- Allows actions to be executed from a list (internal workflow, scheduled commands, operations on a set of objects, etc.), while remaining within the Unified Index framework and controlled behavior.
- Designed for tight interoperability with the Tiki Scheduler, to automate certain operations.
- Also benefits from optimized adaptation of screens and interfaces to offer clear, secure actions to users.

This refactoring has several advantages:

- Offload searches to the indexing engine (which is specialized and optimized for that).
- Drastically reduce the number of heavy SQL queries executed by MariaDB.
- Better control which fields are actually loaded and displayed.
- Better withstand abnormal loads generated by bots.

The concrete result, measured via Zabbix and server tools, is a significant decrease in CPU and memory usage by MariaDB and PHP-FPM workers on these pages, even under heavy crawler pressure.

Limiting retrieved fields to reduce memory usage (from Tiki29 onwards)

Advanced Tiki installations (intranets, business portals, knowledge bases) can end up with indexes containing thousands of Tracker fields. In this context, each query that “retrieves everything” becomes expensive, especially if you perform calculations or subtotals on a large number of results.

New "fields" parameter in lists (select context)

Starting with the Tiki 29 version, an enhancement commit adds the ability to explicitly select which fields to retrieve in the context of a List / CustomSearch plugin, in order to reduce memory usage.

A few key points:

- Only the fields listed in `fields` will actually be retrieved from the search index.
- This capability is built on Tiki's unified indexing layer, common to the three supported engines (MySQL, Elasticsearch, Manticore).(Tiki4)
- The technical fields `object_type` and `object_id` are always available, even if they are not mentioned.

This optimization offers many benefits:

- You only retrieve three business fields (title, status, owner),
- All other tracker fields remain in the index but are not loaded into memory for this list,
- On an index with thousands of fields and tens of thousands of items, this reduction can make a significant difference in terms of RAM usage and processing time for calculations and subtotals.

The same logic applies in contexts where CustomSearch relies on the Unified Index to run complex queries.
If you already use the List or CustomSearch plugins but are not taking advantage of this parameter, OpenSource Solutions can analyze your existing Tiki pages, identify the most resource expensive usage for lists, correctly implement `fields`, and optimize your screens. We can also upgrade your Tiki to a suitable version so you can benefit from these new features in stable and secure conditions. You can entrust us with this targeted refactoring to sustainably reduce resource consumption on your Tiki Wiki CMS.

A summary of a proactive approach for your Tiki Wiki

Against increasingly aggressive AI crawlers and bots, defense cannot be limited to the firewall or the web server. It is crucial to align:

1. The monitoring layer

- Matomo to see *who* is consuming *which pages* and *how*,
- Zabbix to see *how the server is coping* (CPU, memory, disk, network).

2. The server configuration

- PHP-FPM sized correctly,
- MariaDB optimized for your actual usage,
- A search engine adapted to your data volume and type (MySQL, Elasticsearch, Manticore).

3. The Tiki architecture
    - Permissions reviewed to limit what bots can trigger,
    - Refactoring of the heaviest pages (Trackers → List / CustomSearch / ListExecute),
    - Systematic use of new capabilities like `fields` to reduce memory used by the Unified Index.

In this way, your Tiki does not merely “survive” AI bots: it remains fast, available, and under control, even when crawlers go wild.