At this point, no one will attempt to deny that tiki uses too much memory and usually takes a long time to generate pages. The problem comes from the fact that tiki is monolithic and multiple variables are being initialized even if they will never be used and a lot of code is being processed for no reason. This page lists known problems that could be adressed to improve Tiki's speed and memory usage.

Permissions

Tiki has a great granularity on how the permissions are being granted. The problem with them is that they are all loaded on every page. tiki-setup.php (I think that's where it relies) declares a variable for every single permission that can be used in tiki.

Instead of systematically declaring all variables, the pages should define which permission they can use. This way, the permissions concerning the forums would not be loaded in galleries.
lphuberdeau

I think this is a minor hit for performance (we have about 174 permission) that can't be solved realistically. Thinks of the modules on each page, the plugins used, etc.
Chealer9

Features

The problem is very similar. A variable is declared for all possible features. These do make changes on nearly every page since they define which menus will be displayed and multiple other things. Only declaring the enabled ones and using isset() to verify if they are enabled would probably reduce the used memory.

Not a bad idea, but seems only potential for a minor memory gain and needs changes everywhere in the code...probably not an environmentally good solution.
Chealer9

Smarty

$smarty->assign() sequences are way too long. A cache verification before performing the assign sequence could probably help. Instead of populating all feature variables the menu will need, it would be a good thing if the menu will be re-generated in the first place. I'm not a Smarty expert (even part of the opposition), but method calls are not the fastest thing in PHP (you can hear Ilia Alshanetsky ranting about it on the DVD) and they should be reduced. It's not usually a problem, but due to the size of Tiki, it became one.

Also, as a general idea, the permissions passed to Smarty should also be reduced. Instead of sending p_admin, p_wiki_admin, p_edit, p_wiki_edit and all those permissions separatly (random and unverified example), the condition should be performed in PHP and the result only should be sent: can_edit. It also has the advantage of reducing the amount of validations in the template... which is only supposed to be display logic and the amount of differences between templates. In fact, building a permission tree would make the entire thing a lot cleaner.

Database Access

ADODB has the advantage of supporting all databases with a single implementation, but it has the downside of taking all implentations to the lowest of them all in terms of standards. MySQL is the most frequently used database with Tiki and it's also the database with the less standard support. It does have a bunch of specific features that can make it extremely powerful, but using ADODB does not allow to use them.

The solution is not to remove ADODB, it has advantages that cannot be denied, but database server

specific code should be written for critical elements. ADODB would offer a generic approach for all databases that do not have a specific implementation, but MySQL could use it's own features. The FULLTEXT indexing is very powerful and it should be used when available, even if only MySQL has it. The same way databases supporting subqueries should use them if it can reduce the amount of queries sent to the server.

Code Inclusion

Simply because every time a file is included, PHP has to call the file on the filesystem, parse it and convert it to bytecode (which eats up memory), the amount of included files included on general pages should be reduced as much as possible. The filesystem is always the bottleneck. Using an opcode cache solves the problem partially and should be recommanded. (TikiBoosting)

Tiki uses a lot of libraries, it's necessary to make sure they are not always loaded.

According to what I heard and by reading the code, this is the worst problem in Tiki. tiki-setup_base.php, included by almost any Tiki page, includes loads of code, including the well-known monster library tikilib. More information about this problem is available at various places.
Chealer9

---

lphuberdeau
tikilib definetly needs to be splitted in smaller libraries.

HTTP Headers

A good way to leave more processing power to the server is to recieve less requests. Most (If not all) browsers use conditional requests and negociation. As a default value, PHP does nothing with those since the page is supposed to be dynamic. Verifying those headers and indicating to the browser that the page has not changed could cut down the amount of requests. It's necessary to verify those before starting too much libraries or declaring loads of variables, otherwise, there is no gain.

Documentation about conditional headers

Wiki (WikiSyntax)

The wiki has a lot of plug-ins and syntaxes. It could be good to separate them in smaller groups that can be enabled or disabled. The wiki syntax parsing contains way too many regular expressions. Making sure they are not all executed could speed up the parsing.

Some of them are probably very simple and could be replaced by string functions, which are much faster.

Translation

The mechanism used for translation is very simple and eats considerable memory in the context of a feature-bloated monolithic CMS like Tiki. The size of language.php, storing the array used for translation, is in the order of 260 kB. This problem is detailed at I18nDev.