UserPagepacoit

Tiki Performance

I wonder what performance issues the development team has identified and plans to work on in the near future? Following are suggestions of ways to dramatically increase Tiki performance:

Reduce unnecessary code execution per request — Currently, as the developers are surely well aware, Tiki loads a large amount of code just to show, e.g., tiki-index.php. On Tiki 1.7, 74 files totaling about 1MB are loaded. These represent (more or less) the default included code in each request. Much of this overhead can be eliminated by:

1. use conditional includes more agressively
2. separate large language files into smaller (module related?) files.

Call cache check early — Smarty initializes most of itself on every request, even if the page is cached and Smarty only has to resolve the name of the cache file. In fact, currently when cache is turned on, there is only a 20% speed improvement (using phpa); and the same 74 files plus some additional cache files are loaded. Instead of a multiple-fold increase in speed, code initialization (I presume) continues to consume execution time.

1. Modify Tiki to check cache at the earliest possible point.
2. Modify Smarty to initialize and execute only its cache code before initializing the rest.
3. Or, write a separate cache check routine so that Smarty doesn't have to be altered.

Break large files into smaller chunks — *TimeZone.php* and other */lib/Date/* files could be broken up and called conditionally, etc..

Reconsider Tiki object setup code — Hopefully, it will be possible to have much of this code initialized on a far more conditional basis.

Future: High performance and scalability

Thoughts on turning tiki into a "website compiler engine": A means for generating completely static or nearly static pages, whereby no code is executed (not even a cache check) or minimal code is executed on each request.

Imagine for a moment each page with custom written code to run it and only it. Fast, but hoplessly impractical? A backward step to first generation websites? Well, normally, yes. But suppose your CMS took care of recompiling pages according to configuration settings, permissions, etc., every time page content or settings are modified. The impractical grunt work is handled by the CMS. Rather than have the CMS alter settings in the database, why not have it alter the production page directly? That way, each page operates standalone, only initalizing and executing code for its current configuration. And for most pages on most websites, the pages can be 100% static. Website "compilation" would provide huge scalability---and a huge cost reductions for webhosting services specializing in tiki hosting.

Topic Maps for rich navigation

...See TopicMapDev