1 Preventive Measures for Performance Enhancement

1.1 Description

This chapter explains what should or should not be done in order to keep the performance high

1.2 Memory Leakage

Memory leakage in 1.0 docs.

1.2.1 Proper Memory Pools Usage

Several mod_perl 2.0 APIs are using Apache memory pools for memory management. Mainly because the underlying C API requires that. So every time Apache needs to allocate memory it allocates it using the pool object that is passed as an argument. Apache doesn't frees allocated memory, this happens automatically when a pool ends its life.

Different pools have different life lengths. Request pools (`$r->pool`) are destroyed at the end of each request. Connection pools (`$c->pool`) are destroyed when the connection is closed. Server pools `$s->pool`) and the global pools (accessible in the server startup phases, like `PerlOpenLogsHandler` handlers) are destroyed only when the server exits.

Therefore always use the pool of the shortest possible life if you can. Never use server pools during request, when you can use a request pool. For example inside an HTTP handler, don't call:

```
my $conf_dir = Apache::Server::server_root_relative($s->pool, 'conf');
```

when you can call:

```
my $conf_dir = Apache::Server::server_root_relative($r->pool, 'conf');
```

Of course on special occasions, you may want to have something allocated off the server pool if you want the allocated memory to survive through several subsequent requests or connections. But this is normally doesn't apply to the core mod_perl 2.0, but rather for 3rd party extensions.

1.3 Maintainers

Maintainer is the person(s) you should contact with updates, corrections and patches.

- Stas Bekman <stas (at) stason.org>

1.4 Authors

- Stas Bekman <stas (at) stason.org>

Only the major authors are listed above. For contributors see the Changes file.

Table of Contents: