

# Accelerating Rails with lighty

**Jan Kneschke**

jan@kneschke.de

RailsConf 2006  
Chicago, IL, USA



# Who is that guy ?

- Jan Kneschke
  - Main developer of lighty
  - Works at MySQL AB
  - Lives in Kiel, Germany
  - Had to choose between Woldcup + Kiel Week + Midsummer and RailsConf
  - Cares about performance
  - Is not a web-designer



# Who are you ?

- Raise you hand
- Who uses lighttpd ?
- Who runs more than 1, 5, 10, 50 lighttpd installations ?
- What about Load Balancing, Static File transfers ?
- mod\_secdownload, mod\_flv\_streaming ?



# Some History

- **Jan 2003:** lighttpd got started as a POC of the C10k-problem
- **Mar 2003:** FastCGI + load-balancing was working
- Outperformed Apache and thttpd from the start
- **Dec 2003:** new configfile format inspired by Perl, PHP, ... array syntax
- **Aug 2004:** conditionals
- **May 2006:** Netcraft reports 63.000 users



# Configfiles

- Basic types: int, string, assoc array
- Conditionals: string compare, regex

```
$HTTP["url"] =~ "^/app1/" {  
    server.document-root = "..."  
    url.access-deny = ( "~", ".inc" )  
}
```



# Architecture

- C10k is short for "**How to handle 10.000 parallel connections on a single host**"
- Single Process, Single Thread, Non-blocking IO
- Optimized Core
  - Event-handling uses OS specific sys-calls like `kqueue()` or `epoll()`
  - Utilizes zero-copy TCP with the help of `sendfile()` and friends



# Architecture

- Idling Connections are cheap
  - Only a few kBytes and one file-descriptor
  - Use Keep-Alive Connections
  - AJAX likes Keep-Alive
  - COMET builds on Keep-Alive
- Idling Connections spend no CPU time
  - `kqueue()` on \*BSD, `epoll()` on Linux and `/dev/poll` on Solaris are  $O(1)$  functions

# Dynamic Content

- Basic rule: offload all CPU intensive work into a separate process
- Choose one of FastCGI, CGI, SCGI, Proxy
  - Separate Privileges between webserver and application, chroot
  - Scale by CPU-core
  - Remote backends
- Don't use mod\_ssi (see first point)



# CGI

- Common Gateway Interface
- Spawns a process on each request
- In 99% of the cases slower than the other options
- If the process is small and you have to run 5000 connections at the same time, it wins again
- Examples: [www.meebo.com](http://www.meebo.com)



# FastCGI

- Mainly for Rails and PHP support
- Rails adopted lighty for its superior FastCGI support
- Installation on application side might be tricky
- License problems
- Very powerful (keep-alive, multiplexing, out-of-band communication)



# SCGI

- A slimmed down version of FastCGI
- Used by the Python community
- Splits the HTTP-Request into header and body and adds a length fields for both parts
- Might lead to easier parsing in contrast to pure HTTP
- Zed Shaw wrote a Rails dispatcher (and doesn't like it anymore)



# Proxy

- A HTTP proxy module with load balancing
- Fair, RR and Hash (CARP)
- The interface to mongrel
- One of the weaker parts of lighty
  - No output streaming
  - Fail-Over handling is missing
  - Balancers broken in 1.4.11 (fixed in SVN)



# Accelerating Rails

- The less Rails you use the faster it gets
  - Cache as much as you can
  - Use the 404-error-handler to call Rails only for non-cached content
  - Use Content-Length, If-Modified-Since and ETag for dynamic content
  - Use CML



# HTTP Caching

- Make your content cachable in the browser by generating ETag headers
- If the generated ETag is equal to If-None-Match: ... send 304 Not Modified
- Cached Rails Content is automatically compressed and ETag'ed by lighty (if properly configured)



# Load Balancing

- FastCGI, SCGI and proxy support load balancing

```
fastcgi.server = ( ".fcgi" => (  
  ( "host" => "10.0.10.1", ...  
  ( "host" => "10.0.10.2", ...
```

- Example: for 10 hosts: Use 1 lighty and 9 rails backend servers



# Load Balancing

- NFS to share the directories
- a shared session storage (like memcache)
- Use graceful restart to have a low impact in the running requests
- Use a TCP multiplexer like (LVS) to add and remove backends at runtime



# Special Modules

- mod\_secdownload
  - Use Rails to authenticate static file transfers
- mod\_flv\_streaming
  - Build your own video.google.com or youtube
- mod\_fastcgi + x-sendfile
  - Let lighty send static files, it is made for it



# mod\_secdownload

- Move sensitive files out of the document-root
- Uses temporary URLs
  - `../download/<md5>/<timestamp>/file`
- `<md5>` is built on the filename, the timestamp and a secret
- `<timestamp>` is a UNIX timestamp for timeout



# X-sendfile

- Authenticate in the App (Rails, PHP, ...)
- Off-load static file transfers to lighty

```
X-sendfile: /data1/files/secret.xls
```

- No extra buffering
- Backend is released earlier
- In 1.4.x only in mod\_fastcgi



# mod flv streaming

- **FLV** are Flash Video files
- The netstream object in Flash can only stream, can't seek
- *Workaround*: create FLVs at runtime by seek into the FLV and prepend a new FLV header
- <http://jan.kneschke.de/projects/flv-streaming/>
- Youtube.com uses lighty, but not flv-streaming yet

# CML

- Short for "**C**ache **M**eta **L**anguage"
- *Assumption:* > 95% Cache Hits
- How to know if we have Hit or Miss ?
- Cost of a Cache Hit is mostly around making the decision
- Move the decision out of the Application
- Make the decision simple



# CML

- Will become a general purpose scripting interface into the server
- Uses LUA as the language
  - Easy to learn, fast, compilable, embedded
  - Portable, small, extensible
- Replacement for RewriteCond in Apache's mod\_rewrite



# Debugging

- Trace the progress of the request handling

```
debug.log-request-handling =  
  "enable"
```

- How many backends are used at the same time

```
status.statistics-handler =  
  "/server-stats"
```



# A look into the future

- lighttpd 1.5.x is on the horizon
- Unified mod\_proxy
- New IO sub-system
- Make mod\_cml mainstream
- Native win32 implementation



# Unified Proxy Module

- `mod_proxy = mod_fastcgi 1.3.x` - FastCGI
- `mod_proxy` has 3 load balancers, `mod_fastcgi` has the fail-over handling
- Unifying all external mods into a `mod_proxy_core` and several protocol implementations
- Use `mmap()` or `vmsplice()` to get zero-copy TCP for `mod_proxy`



# New IO Sub-system

- Streaming input and output
- Filtering of streams (mod\_uploadprogress, mod\_deflate, mod\_layout, ...)
- COMET (aka mod\_multiplex)
- Specialized network handlers for read()ing



# And if you have a wish ?

- HTCP 0.0 (RFC 2756) support for caching remote backends
- [ fill in your wish here ]



# If I have a wish ...

- For lighty
  - A WebUI for the configuration
  - A graphing app for mod\_status
- For Rails
  - Callgrind output from the ruby profiler (even PHP has that via ext/xdebug)
- For me
  - ... a beer



File View Go Settings Help

Time

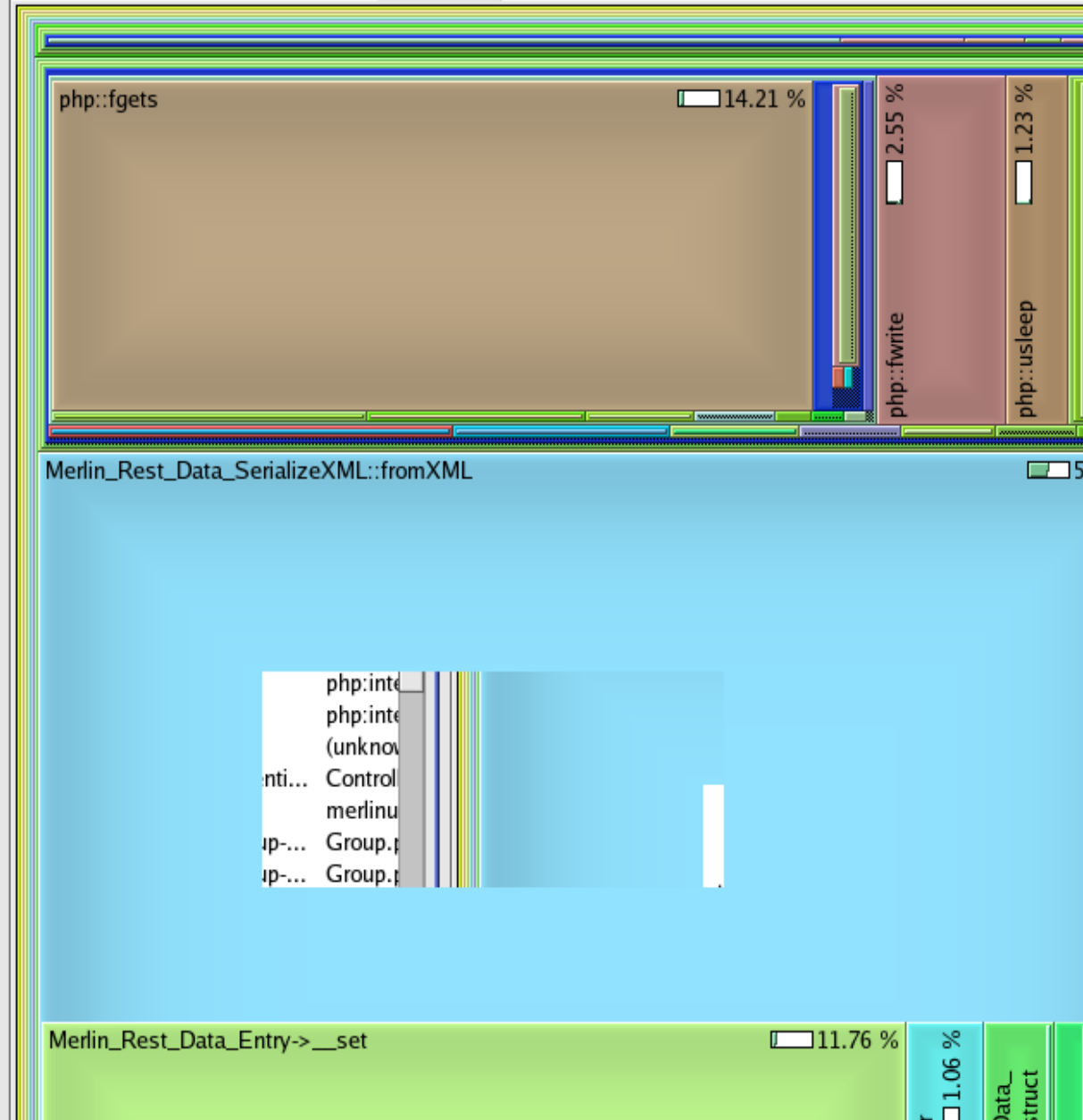
Flat Profile

Search: (No Grouping)

Incl.	Self	Called	Function	Location
100.03	0.28	(0)	{main}	index.p
93.52	0.03	20	Merlin_Controller->dispatch	Control
90.17	0.01	20	php::call_user_func_array	php:inte
90.16	0.37	19	Merlin_Controller_Events->in...	Events
77.96	0.53	19	Merlin_Model_Events->getAll	Events
73.91	0.08	19	Merlin_Rest_Service_Monitor...	Monitor
51.02	35.50	9 168	Merlin_Rest_Data_SerializeX...	Serializ
38.19	0.11	181	Merlin_Rest_Service->send	Service
38.07	0.51	181	Merlin_Rest_Service->post	Service
36.85	1.21	181	Merlin_Http->post	Http.ph
26.28	0.90	181	Merlin_Http->_get_response	Http.ph
22.84	22.84	1 248	php::fgets	php:inte
11.99	11.44	40 360	Merlin_Rest_Data_Entry->__set	Entry.p
11.31	0.13	19	Merlin_Model_Server->getTree	Server.
9.75	0.10	104	Merlin_Rest_Service_Group-...	Group.p
5.05	0.16	22	require_once::/home/jan/proje...	merlinu
4.79	4.79	22	php::session_start	php:inte
4.10	4.10	362	php::fwrite	php:inte
3.67	2.81	411	<cycle 1>	(unkno
3.30	0.01	20	Merlin_Controller->isAuthenti...	Control
2.59	2.07	451	__autoload <cycle 1>	merlinu
2.10	0.01	19	Merlin_Rest_Service_Group-...	Group.p
2.09	0.03	19	Merlin_Rest_Service_Group-...	Group.p
2.09	0.05	67	Merlin_Model_Factory::create	Model.p
2.08	0.02	19	Merlin_Rest_Service_Monitor...	Monitor
1.97	1.97	181	php::usleep	php:inte
1.90	1.67	5 411	Merlin_Http_Header->_format...	Header
1.84	0.03	20	Merlin_Rest_Service_Securit...	Securit
1.55	0.59	3 058	Merlin_Http_Header->set_he...	Header
1.50	0.67	181	Merlin_Http_Response_Head...	Header
1.49	0.35	1 086	Merlin_Http_Response_Mess...	Messag
1.47	0.52	2 715	Merlin_Http_Header->get_he...	Header
1.22	1.12	9 168	Merlin_Rest_Data_Entry->__...	Entry.p
1.14	1.14	130 926	php::strtolower	php:inte

{main}

Types Callers All Callers Source Callee Map



# Thanks

## Getting Help:

#lighttpd at irc://freenode  
mailto:lighttpd@lists.kneschke.de  
catch me while I'm here

