Advanced PHP, Apache, and MySQL Settings

Going Beyond the Basics

Mario Peshev

Technical Trainer

www.peshev.net

Software University

http://softuni.bg

Table of Contents

SOFTWARE UNIVERSITY FOUNDATION

PHP Settings

The `php.ini` file

PHP Settings

- PHP settings (called directives) are stored in the **`php.ini`** file
  - A set of **`name = value`** statements
    - If there is no value, the directive is left as **`name =`**
    - Comments start with a semicolon
  - The location of the file is different across operating systems and versions
  - You can check which **`php.ini`** file is loaded with **`phpinfo()`**
- PHP supports add-ons
  - Most add-ons read their settings from the same file

Language Settings

Directive	Default	Description
short_open_tag	Off	Whether <? and ?> tags should be allowed
asp_tags	Off	Whether <% and %> tags should be allowed
precision	14	The number of significant digits displayed in floating-point numbers
expose_php	On	Include X-Powered-By: PHP/<version> HTTP header in responses
disable_functions	""	A comma-separated list of which functions should not be available at runtime
disable_classes	""	A comma-separated list of which classes should not be available at runtime

Data Handling Settings

Directive	Default	Description
`track_vars`	`"On"`	Whether or not the EGPCS ($_ENV, $_GET, $_POST, $_COOKIE, $_SERVER) arrays will be available
`register_globals` (Removed in PHP 5.4.0)	`Off`	Whether or not to register the EGPCS arrays as global variables (e. g. `$_GET["name"]` will become `$name`)
`variables_order`	`"EGPCS"`	The order in which to register the global EGPCS arrays ("GP" means that only $_GET and $_POST will be available)
`post_max_size`	`"8M"`	The maximum size of post data allowed
`default_mimetype`	`"text/html"`	The default MIME type of the HTTP response
`default_charset`	`"UTF-8"`	The default charset of the HTTP response

File Upload Settings

Directive	Default	Description
`file_uploads`	`"On"`	Whether or not to allow file uploads
`upload_tmp_dir`	`NULL`	The temporary directory to store files at the server during upload
`upload_max_filesize`	`"2M"`	The maximum size of an uploaded file
`max_file_uploads`	`20`	The maximum number of files allowed to be uploaded at the same time

Paths and Directories

Directive	Default	Description
include_path	".;path/to/php/pear"	The path where functions like include() and require() look for files
open_basedir	NULL	The only base directory (and all its subdirectories) which are accessible to PHP
doc_root	NULL	PHP's "root directory" on the server
user_dir	NULL	The user's home directory for PHP files
extension_dir	"path/to/php"	The path where PHP should look for dynamically loadable extensions

Other Settings

Directive	Default	Description
output_buffering	"4M"	How much output should be kept before pushing it to the client ("On" for unlimited buffer side, integer value for buffer size)
implicit_flush	Off	Whether or not to flush the output buffer after each block
max_execution_time	120	Maximum execution time of a script in seconds
memory_limit	"128M"	Maximum memory (in bytes) that a script is allowed to allocate (-1 for no memory limit)

Changing Settings at Runtime

- Get the runtime value of a `php.ini` variable:

```
ini_get("upload_max_filesize");
```

- Get the value of a `php.ini` variable which is stored in the file:

```
get_cfg_var("upload_max_filesize");
```

- Change the value of a `php.ini` variable at runtime:

```
ini_set("include_path", "c:/php/PEAR");
```

- See the current values of the PHP settings:

```
phpinfo();
```

SOFTWARE UNIVERSITY FOUNDATION

Apache Settings

The `httpd.conf` file

Apache Settings

- Apache settings are defined in the **`httpd.conf`** file
  - Location and name may differ across platforms and Apache versions
  - Older version read from multiple files
  - The site-specific settings and module-specific settings are in separate files
  - Follows a syntax close to XML
    - Name-value pairs sometimes in tags

Prefork vs. Worker

SOFTWARE UNIVERSITY FOUNDATION

- Apache has two core modules (versions) – prefork and worker
  - Different behavior
  - Prefork is process-based, does not utilize threads much, better for single / dual core CPU servers
  - Worker utilizes threaded architecture – better for multi-core / CPU servers
  - The two modules perform differently on different tests

Apache Modules

- Load a module:

```
LoadModule ssl_module modules/mod_ssl.so
```

- Use conditional configuration settings:

```
<IfModule dir_module>
    DirectoryIndex index.php
    DirectoryIndex index.html
</IfModule>
```

- Load mod_php:

```
LoadModule php5_module “C:/Program Files/PHP/php5apache2_2.dll”
```

Connection Settings

Directive	Default	Description
`TimeOut`	`300`	The number of seconds before the server sends timeout to a dead connection
`KeepAlive`	`On`	Turns persistent connection on or off
`MaxKeepAliveRequests`	`100`	The maximum number of persistent connections allowed at the same time
`KeepAliveTimeout`	`5`	The number of seconds before closing a dead persistent connection

Log Settings

Directive	Default	Description
`ErrorLog`	`logs/error_log (Linux) / logs/error.log(Windows)`	Sets the Apache log file; can be specified separately for each site
`LogLevel`	`warn`	Sets the level of logging; possible values – `debug`, `error`, `info`, `notice`, `warn`, `error`, `crit`, `alert`, `emerg`
`LogFormat`	`"%h %l %u %t \"%r\" %>s %b"`	Specifies nicknames for different log formats; can be used for site-specific access logs

Other Settings

Directive	Default	Description
`Listen`	`80`	Sets the port to listen for connections; can be repeated with different ports; usually specified in `ports.conf` file
`HostnameLookups`	`Off`	If turned on, logs the host names of remote clients instead of IP addresses
`User / Group`	`#-1`	Sets the user and group that the Apache process should work in
`DirectoryIndex`	`index.html`	Sets the default files in a directory (shown when the user requests the directory)

Virtual Hosts

- Apache supports multiple sites on the same IP address / port
  - Specified in **`VirtualHost`** directives
  - Usually virtual hosts are separated in different files
  - Requires **`NameVirtualHost`** directive
    - Sets the IP address and port on which Apache will receive requests for the name-based virtual hosts
    - IP and Port can be replaced with `*` (any)

Virtual Host Example

```
NameVirtualHost *:80

<VirtualHost *:80>
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example/htdocs
    ErrorLog /var/www/example/logs/err
    CustomLog /var/www/example/logs/custom
</VirtualHost>
```

- **ServerName** specifies the domain name of the virtual host
- **ServerAlias** specifies additional names (domains) for this virtual host

Virtual Host Example (2)

- **`DocumentRoot`**
  - Sets the root directory for this host
  - Passed to PHP in the **`$_SERVER["DOCUMENT_ROOT"]`** variable
  - Be careful with the ending slash!
- **`ErrorLog`** sets the host-specific error log
- **`CustomLog`** sets the location and format for the host access log file

Location Directive

```
<VirtualHost *:80>
    …
    <Location /admin>
        Require valid-user
    </Location>
</VirtualHost>
```

- The **`Location`** directive is used to define URL-specific settings
  - Settings are directory-based
  - Can be placed in **`VirtualHost`** or be server-wide

Directory Directive

```
<VirtualHost *:80>
    …
    <Directory /var/www/includes>
        Allow from localhost
        Deny from all
    </Directory>
</VirtualHost>
```

- The **`Directory`** directive is used to define file system directory settings
  - Can be defined server-wide or host-specific

Deny and Allow

- The **`Deny from`**, **`Allow from`** and **`Order`** directives are used to limit access to certain hosts
  - **`Deny`** and **`Allow`** values are lists of hosts (space-separated), partial domain names, partial IPs or "all"
  - The **`Order`** directive sets whether deny or allow should be higher priority
    - Value is **"`Allow,Deny`"** or **"`Deny,Allow`"**
    - The first is with higher priority, if a host is not matched, the second in list is used

Deny and Allow – Examples


```
Allow from localhost
Deny from all
Order Allow, Deny
```

```
Allow from .net # partial domain
Deny from 192.168 # partial IP
Order Deny, Allow
```

```
Allow from localhost 192.168.0.1
Deny from 85.187.0.0/16 # deny a network
Order Deny, Allow
```

```
Allow from 2001:db8::a00:20ff:fea7:ccea
Deny from all
Order Allow, Deny
```

The Options Directive

- Sets values of several additional directory-based options
  - Each option is prefixed with + or – to turn `on` or `off`; if no prefix is supplied, `on` is assumed
  - **`ExecCGI`** – whether CGI scripts execution is allowed in the directory
  - **`FollowSymLinks`** – whether Apache should use only files or can follow symbolic links in the directory

The Options Directive (2)

- **`Indexes`** – If a URL maps to directory and there is no file that matches the **`DirectoryIndex`** directive then **`mod_autoindex`** will return page with the list of files in the directory
- Turning this on for hosts / locations that do not explicitly require it is considered security risk!

```
<Directory /var/www/docs>
    Options +Indexes +FollowSymLinks -ExecCGI
</Directory>
```

Setting up a Virtual Host – Example

- To set up a virtual host:

1. Set your domain name to point to your external IP address
   - For testing you can modify the `hosts` file
     - **`/etc/hosts`** on Linux
     - **`C:\WINDOWS\System32\drivers\etc\hosts`** on Windows
2. Add **`NameVirtualHost`** and **`VirtualHost`** directives in **`httpd.conf`**
3. Restart Apache

Using HTTPS

- **HTTPS** is **HTTP** over SSL / TLS
- Apache has separate module for handling **`https`**
- Running virtual host over **`https`** requires a certificate and connection on port 443
  - In Linux the packages **`openssl`** and **`ssl-cert`** are necessary too
  - Apache has automated script for generating certificates – **`apache2-ssl-certificate`**

Configuring HTTPS

- Example of virtual host with SSL:

```
<VirtualHost *:443>
    ServerName phpmyadmin.example.com
    DocumentRoot /usr/shared/phpmyadmin/
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/myadmin.pem
</VirtualHost>
```

- The **`SSLEngine`** directive turns on the SSL security engine
- **`SSLCertificateFile`** supplies a valid certificate file
  - The domain property in the file must match the host name

Configuring HTTPS – Example

1. First ensure that **`httpd-ssl.conf`** file will be loaded. Put this code in **`httpd.conf`**:

```
Include conf/extra/httpd-ssl.conf
```

2. Create a self-signed SSL certificate:

```
openssl genrsa 1024 > host.key
```

```
openssl req -new -x509 -nodes -sha1 -days 365 -key host.key > host.cert
```

```
cat host.cert host.key > host.pem
```

3. Define a virtual host on port 443 with SSL engine switched on
4. Restart Apache

HTTP Authentication

- The Apache module **`mod_auth`** allows the use of basic HTTP authentication
  - Restrict or allow access to certain areas
    - Requires user and password input
  - For stronger authentication and scalability use **`mod_auth_digest`** or **`mod_auth_dbm`**
  - Usernames and password are stored encrypted in a file

mod_auth Directives

Directive	Description
`AuthType`	Sets the type of user authentication; possible values – `Basic` and `Digest`
`AuthName`	User-friendly name of the realm that requires authorization; must be enclosed in quotation marks
`AuthUserFile`	Specifies the file which stores users and passwords
`AuthGroupFile`	Specifies the file which stores the user groups (lists of users); groups cannot be nested or inherited

- Example content of a **groups** file:

```
Boss: john pesho
Accounting: mara ceca
Testers: chocho bobo shusi
```

- Never put the **users** or **groups** files in the document tree of the site!

Require Directive

- **`Require`** sets which users/groups are allowed to access the realm
  - The possible values are:

```
Require user [list of users]
Require group [list of groups]
Require valid-user
```

The htpasswd Tool

- Apache comes with a small tool for generating user files named **htpasswd**
  - Encrypts the passwords
  - Usually these files are named **.htpasswd**

```
// The –c flag means “create a new file”
htpasswd –c .htpasswd mara
// Asks you to supply password

// Add new user
htpasswd .htpasswd john
// Again asks for password
```

Authentication – Example

```
<VirtualHost *:80>
    ServerName example.com
    DocumentRoot /var/www/ex/htdocs
    …
    <Location /admin>
        AuthType Basic
        AuthName "Example admin area"
        AuthUserFile /var/www/ex/.htpasswd
    </Location>
</VirtualHost>
```

Using .htaccess

- Apache can read additional settings from files in the site document tree
  - The name of the file is controlled by the **`AccessFileName`** server directive
  - Usually named **`.htaccess`**
- In the **`.htaccess`** file can be placed all directives, valid for **`Location`**
- Slows down the server
  - It has to read it again on every request

Example .htaccess

```
Options +Indexes
AuthType Basic
AuthName "test"
AuthUserFile ".htpasswd"
php_value register_globals off
```

- Apache reads all **.htaccess** files in the directories from the document root up to the requested resource and combines them
- Can contain **mod_rewrite** settings
- Can contain PHP settings with the **php_value** directive

mod_rewrite

- **mod_rewrite** allows rule-based rewriting and redirecting of requests
  - Example: user requests **index.html** but the rewrite rules change this to **index.php**
  - This is NOT redirecting!
  - Used to make friendly URLs, rename resources, etc.
  - Based on regular expressions
  - Operates on per-server or per-directory context

Rewriting Directives

Directive	Description
`RewriteEngine`	Enables or disables the runtime URL rewriting engine
`RewriteBase`	Sets the base URL for per-directory (`.htaccess`) rewriting
`RewriteRule [pattern][substitution][flags]`	If the requested URL matches the pattern it is rewritten with the replacement; allows using back-references and groups

RewriteRule Flags

- `[L]` – rewriting should stop and no other rules should be checked
- `[F]` – force 403 Forbidden response code
- `[G]` – force 410 Gone response code
- `[R=(code)]` – force redirect with response code
  - User is redirected to the result URL
- `[N]` – restart rewriting with the new address
- `[NC]` – case-insensitive match
- `[C]` – chain the current rule with the next
  - If not matched, skips the chained rules

URL Rewriting – Example

```
RewriteEngine On
# Rewrite directories to index files
RewriteRule ^(.*)/$ $1/index.html

# Send all html files to the template engine
# so the URLs are friendly
RewriteRule ^(.*).html$ /template.php?page=$1

# Generate the human validation image
RewriteRule ^captcha.gif$ /captcha_gen.php

# Stream the videos
RewriteRule ^/(.{10}).swf$ /stream.php?vid=$1

# Rewrite product URLs
RewriteRule ^/products/(.*)/(.*).html$
        /product.php?category=$1&product=$2
```

RewriteCond

- The **`RewriteCond`** directive defines a rule condition
  - Used to match HTTP headers, connection and request properties, server settings, system properties, etc.
  - One or more **`RewriteCond`** directives can precede **`RewriteRule`** directive
    - All conditions must match to rewrite the URL

RewriteCond – Example


```
# Mozilla users special page
RewriteCond ${HTTP_USER_AGENT} ^Mozilla.*
RewriteRule ^/index.html$ /index.mozilla.php

# Internal network special home page
# Use for the 10.0 and 192.168 networks
RewriteCond %{REMOTE_HOST} ^10.0.*$ [OR]
RewriteCond %{REMOTE_HOST} ^192.168.*$
RewriteRule ^/index.html$ /index.internal.php

# Only HTTP authenticated user admin
RewriteCond %{REQUEST_METHOD} ^HEAD$
RewriteCond %{REMOTE_USER} ^admin$
RewriteRule .* $1 [F] # Force forbidden!
```

SOFTWARE UNIVERSITY FOUNDATION

MySQL Settings

The `my.cnf` and `my.ini` files

MySQL Settings

- MySQL settings are in the following files:
  - `my.cnf`
  - `my.ini`
- Split into sections
  - Section name is defined in `[` and `]`
  - Settings are in `name = value` form

Network Settings

Setting	Description
`port`	Sets the connection port (usually `3306`); passed to all clients
`bind-address`	Sets interfaces to listen on; for security reasons this is usually `127.0.0.1` (allows only local connections)

Fine tuning settings

- Fine tuning of MySQL is done in the mysqld section
  - Defines memory usages for buffers and connections

Setting	Description
`key_buffer`	Size of the cache buffer for primary and foreign keys
`join_buffer`	Size of the cache buffer for matching fields from two tables; increase if multiple joins in one query are used often
`sort_buffer_size`	Size of the buffer for sorting; increase when sorting too many rows
`thread_cache_size`	Size of the cache for each thread; increase when running nultiple queries on the same tables in a single script

Fine tuning settings (2)

SOFTWARE UNIVERSITY FOUNDATION

Setting	Description
`table_cache`	Size of the per-table cache
`thread-concurrency`	Concurrency level of threads; supposed to affect only Solaris platforms (but works well on Linux); set to double the number of CPU cores
`wait_timeout`	The number of seconds to wait before closing a dead connection
`wait_interactive_timeout`	The time the server waits for a persistent connection

MySQL Tuning – Example

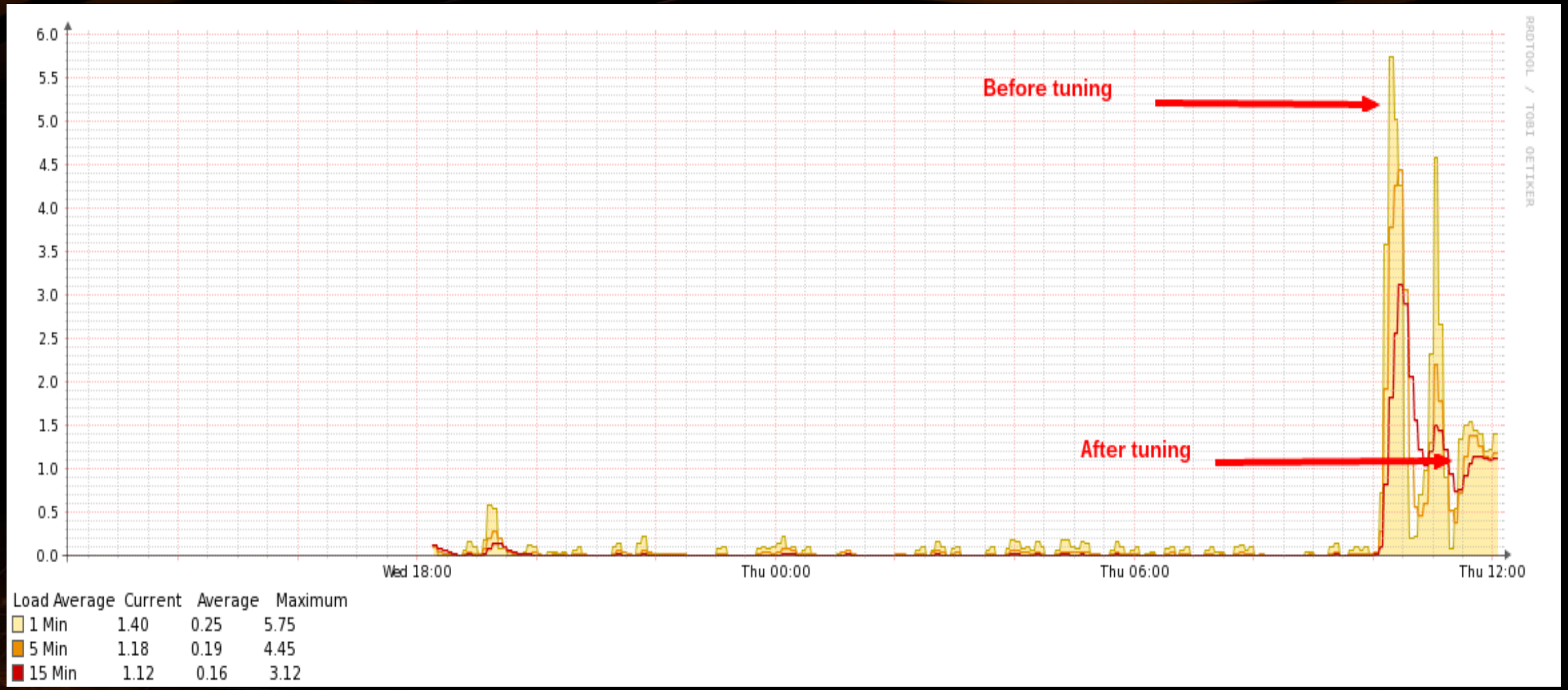
- Always play around with the settings, testing with benchmarks
  - Apache Benchmark (AB)

```
key_buffer              = 250M
max_allowed_packet      = 16M
thread_stack            = 128K
thread_cache_size       = 128
max_connections         = 1000
table_cache             = 6000
thread_concurrency      = 16

wait_timeout            = 100
interactive_timeout     = 100
connect_timeout         = 10
```

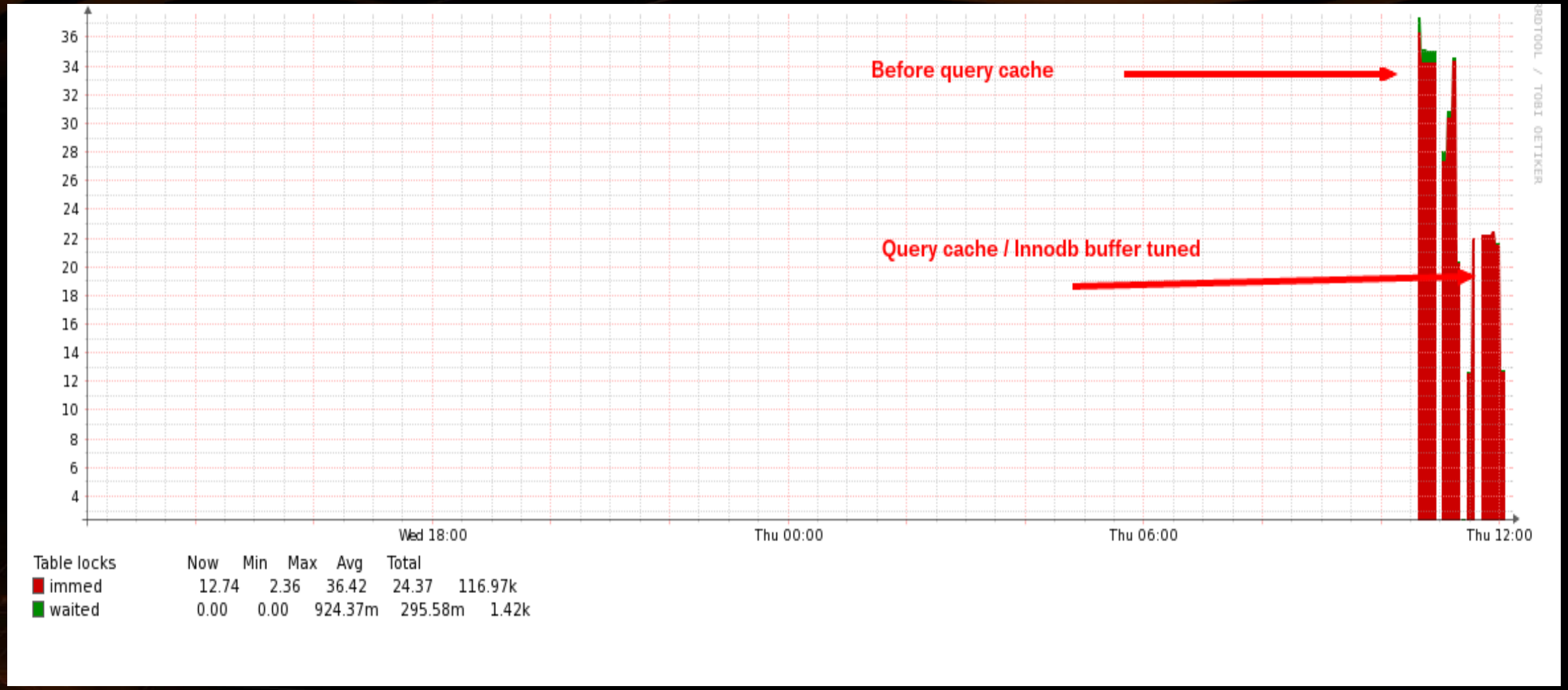
MySQL Tuning – Example (2)

```
join_buffer               = 2M
sort_buffer_size          = 2M
read_buffer_size          = 2M
read_rnd_buffer_size      = 768K
myisam_sort_buffer_size = 64M

query_cache_limit         = 4M
query_cache_size          = 128M
query_cache_type          = 1
```

Result from changing my.cnf / my.ini files



Result from changing my.cnf / my.ini files

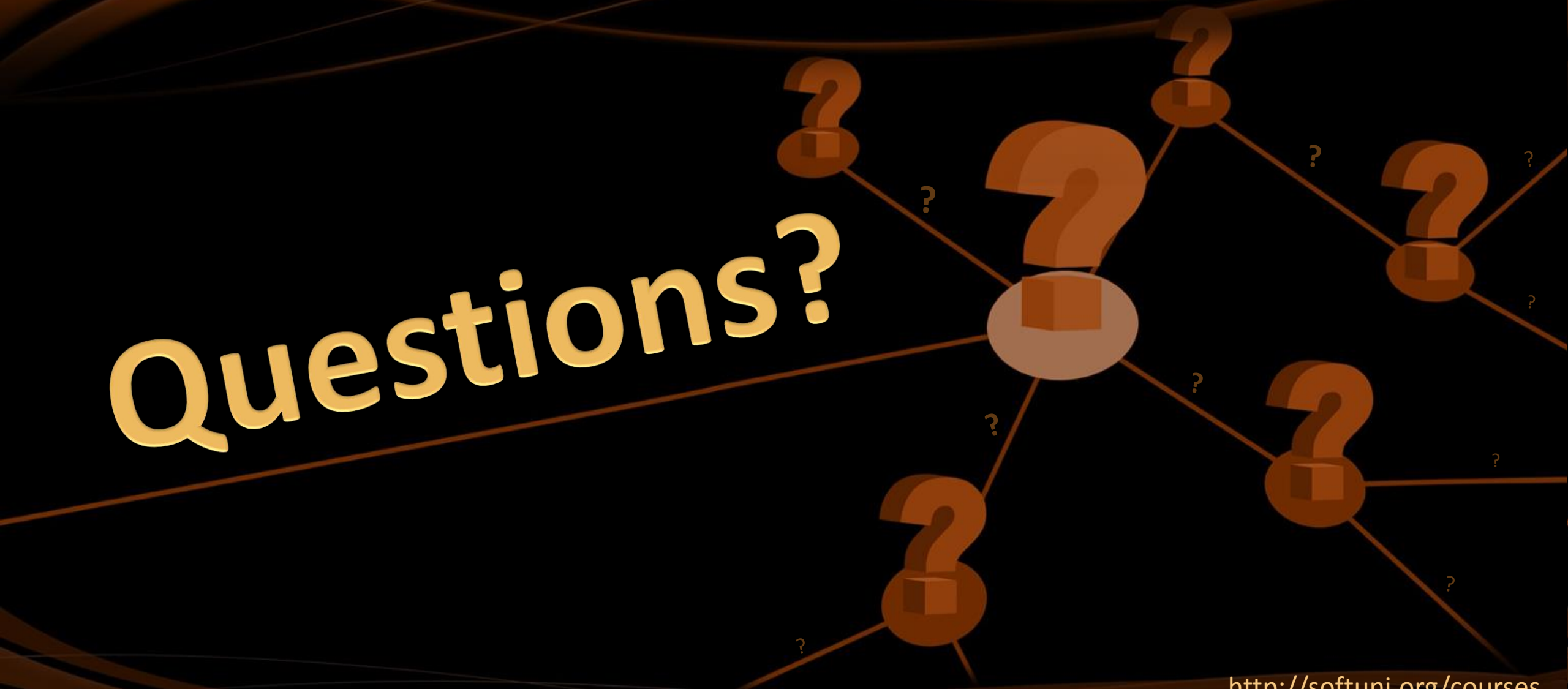


Summary

- PHP Settings
  - **`php.ini`** file
- Apache Settings
  - **`httpd.conf`** file
  - Virtual hosts
  - Modules
- MySQL Settings
  - **`my.cnf`** / **`my.ini`** files
  - Fine tuning and performance

Advanced PHP, Apache, and MySQL Settings

SOFTWARE UNIVERSITY FOUNDATION



License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Fundamentals of Computer Programming with C#" book by Svetlin Nakov & Co. under CC-BY-SA license
  - "C# Part I" course by Telerik Academy under CC-BY-NC-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg