TYPO3 Core - Bug #52949

Epic # 55070 (Closed): Workpackages

Epic # 55065 (Closed): WP: Overall System Performance (Backend and Frontend)

Speed decrease since 4.5

2013-10-18 13:13 - Philipp Gampe

Status:	Closed	**Start date:**	2013-09-19
Priority:	Should have	**Due date:**	
Assignee:		**% Done:**	100%
Category:	Performance	**Estimated time:**	0.00 hour
Target version:			
TYPO3 Version:	6.2	**Complexity:**	hard
PHP Version:	5.3	**Is Regression:**	No
Tags:		**Sprint Focus:**	

Description

There is a very notable decrease in speed since 4.5.

A fully cached request in 4.5

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   3.513 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    28.47 [#/sec] (mean)
Time per request:       35.129 [ms] (mean)
Time per request:       35.129 [ms] (mean, across all concurrent requests)
Transfer rate:          34.42 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:    32   35   3.3     34      59
Waiting:       32   35   3.3     34      58
Total:         32   35   3.3     34      59

Percentage of the requests served within a certain time (ms)
  50%     34
  66%     36
  75%     37
  80%     38
  90%     38
  95%     39
```

```
  98%     45
  99%     59
 100%     59 (longest request)
```

A fully cached request in 6.2beta1

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        752 bytes

Concurrency Level:      1
Time taken for tests:   14.486 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      105300 bytes
HTML transferred:       75200 bytes
Requests per second:    6.90 [#/sec] (mean)
Time per request:       144.856 [ms] (mean)
Time per request:       144.856 [ms] (mean, across all concurrent requests)
Transfer rate:          7.10 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   120  145  16.4    142     202
Waiting:      119  144  16.4    141     201
Total:        120  145  16.4    142     202

Percentage of the requests served within a certain time (ms)
  50%    142
  66%    148
  75%    150
  80%    155
  90%    167
  95%    181
  98%    202
  99%    202
 100%    202 (longest request)
```

Find attached sample xhprof and cachegrind files, plus the used (simple) TS template and some notes for the setup (1 page, no domain record).
Profiling was done with web3tracer.

Subtasks:

Task # 53029: Improve performance of the class loader	**Closed**
Task # 53744: Change ClassLoader cache from proxy require files to standard cache backends	**Closed**
Task # 53747: Change TCA cache from php code to serialized array	**Closed**
Feature # 52642: CF: Implement Simple db, apc & memcached backend	**Closed**
Bug # 53556: classLoader->loadClass calls requireOnce twice per class	**Closed**
Bug # 53918: t3skin calls addIconSprite for each language	**Closed**
Bug # 53962: Class loader does not cache non existing classes	**Closed**
Bug # 53702: FAL fetches storages twice on every BE request	**Closed**
Task # 54251: GeneralUtility::array_merge_recursive_overrule has bad performance	**Closed**

Bug # 53862: isValidUrl idna converts whole URI instead of domain only	**Closed**
Bug # 52125: Saving records takes ages to complete	**Closed**
Bug # 52235: Timeout when copying pages recursively due to cache-clearing overload	**Closed**
Bug # 53598: Select/Delete fe_sessions twice per request	**Closed**
Bug # 54982: Performance of eID-scripts in TYPO3 6.2	**Closed**
Bug # 55022: PackageManager Cache is invalid if typo3 dir has symlinks	**Closed**
Bug # 56307: Findings from Bootstrap/PackageManager/ClassLoader analyses	**Closed**
Bug # 56308: ClassLoaderCache superflouous	**Closed**
Bug # 56310: The PackageManager checks every boot if all required packages are active	**Closed**
Bug # 56311: PackageFactory does not need to be initialized on regular requests	**Closed**
Bug # 56312: Dependency Resolver does not need to be initialized	**Closed**
Bug # 56313: Cache Identifiers shorten the MD5 hash	**Closed**
Bug # 56314: setCacheHashOptions from Bootstrap not relevant for EID	**Closed**
Bug # 56315: Optimize TimeTracking init in eID Case	**Rejected**
Bug # 56316: inline BEUserAuth::getCookieName	**Closed**
Bug # 56317: Cache Local and Default Configuration to prevent duplicate File Access	**Rejected**
Bug # 56318: Check wether the ConfigurationManager can be omitted/reduced for the purpo...	**Closed**
Bug # 56319: Merge Cache/Cache and CacheFactory into CacheManager	**Closed**
Bug # 56320: Check emptying the Flow Classes as far as possible	**Rejected**
Bug # 56322: GetAliasesForClass not used	**Closed**
Bug # 56323: cache_classes should cluster by Package instead of Class	**Closed**
Bug # 56324: Class Loader should not try to runtime resolve class-pathes	**Closed**
Bug # 56325: Move "finalClassNameCache" from GeneralUtility to ClassLoader	**Closed**
Bug # 56326: Evaluate and Merge ClassAliasMap into Classloader class	**Rejected**
Bug # 56328: Extract complex "class loader cache building" code into ClassLoadingInform...	**Closed**
Bug # 56330: Add all classes which are needed anyhow within the minimal Bootstrap to th...	**Rejected**
Bug # 56331: Evaluate if we can profit from concatenate all baseRequiredClasses into on...	**Closed**
Bug # 56333: MicroOptimize ClassLoader	**Closed**
Bug # 56334: Prevent inclusion of LogLevel due to ConfigDefault referral	**Rejected**
Bug # 56335: Prevent Inclusion of ResourceStorage from Bootstrap	**Closed**
Bug # 56336: Prevent Inclusion of MediaWizardProvider* in Base Bootstrap	**Closed**
Bug # 56337: Remove manual array calculation in Locales and put the arrays to the class...	**Closed**
Bug # 56340: Remove the VariableFrontend from requiredBaseClasses	**Closed**
Task # 56341: Evaluate if it is really reasonable to have TYPO3\CMS\Core\Compatibility\...	**Closed**
Task # 56538: Cache the $GLOBALS['TYPO3_LOADED_EXT'] as an array	**Closed**
Bug # 56933: Cached class loader misses are not considered on retrieval	**Closed**
Task # 56934: Move information in Package classes into composer files	**Closed**
Story # 57862: Add possibility to make all TCA additions cached	**Closed**
Task # 57863: Introduce a signal in ExtensionManagementUtility::loadBaseTca	**Closed**
Task # 57881: Make category TCA changes cacheable	**Closed**
Task # 57942: Provide API to add cached TCA changes	**Closed**

Related issues:

Related to TYPO3 Core - Task #52304: Performance issues	**Closed**	**2012-03-15**

Associated revisions

Revision 2bfb2a64 - 2014-02-11 23:27 - Jost Baron

[TASK] Improve EM performance when getting extension list from TER

When importing the extension list from TER, each version of each
extension is one database record. For each extension key the EM
calculates the newest version and sets the column 'current_version'
to 1 for the corresponsing extension record.

The old implementation issued one database query for each extension
key, resulting in about 6k queries, taking a long time to execute
(several minutes on my machine). After applying this patch the same

thing is done in three queries, speeding up the process.

Resolves: #55820
Related: #52949
Releases: 6.2
Change-Id: I55d3699a63b13e7c07af4babbf57d0d06f367027
Reviewed-on: https://review.typo3.org/27496
Reviewed-by: Markus Klein
Tested-by: Markus Klein
Reviewed-by: Jost Baron
Tested-by: Jost Baron
Reviewed-by: Oliver Klee
Reviewed-by: Wouter Wolters
Tested-by: Wouter Wolters

History

#1 - 2013-10-18 13:14 - Philipp Gampe

- File profiles.tar.gz added

#2 - 2013-10-18 13:51 - Philipp Gampe

Full VM for VirtualBox 4.2 (1GB): http://philippgampe.info/fileadmin/ubuntu-typo3.ova.zip

#3 - 2013-10-18 14:34 - Frans Saris

Philipp, did you also test the 4 other versions. Im interested to know if the last changes are the source of the bigger performance lost or that is was something we introduced in a earlier version. Compatibility layer etc.....

#4 - 2013-10-18 16:48 - Philipp Gampe

- File typo3-4-5-fully-cached.png added

- File typo3-6-2beta1-fully-cached2.png added

Will test the other versions later today. There does not seem to be a single bottleneck.

#5 - 2013-10-20 15:37 - Philipp Gampe

I ran all tests at least twice and warmed up the caches with a couple of browser requests beforehand. Thus those numbers represent a fully cached condition.
All tests are on the same virtual machine (see link above), on the same DB and a default set of packages/extensions. See the zip file attached for localconf.php, LocalConfiguration.php and PackageStates.php.

Tests are started from host and query the VM. It can be expected that additional I/O load has a bigger impact than on real-life servers.

TYPO3_4-5

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   1.252 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    79.86 [#/sec] (mean)
Time per request:       12.522 [ms] (mean)
Time per request:       12.522 [ms] (mean, across all concurrent requests)
Transfer rate:          96.55 [Kbytes/sec] received
```

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:    10   12   1.8     12      24
Waiting:       10   12   1.8     12      23
Total:         11   12   1.8     12      24

Percentage of the requests served within a certain time (ms)
  50%     12
  66%     13
  75%     13
  80%     13
  90%     14
  95%     16
  98%     20
  99%     24
 100%     24 (longest request)
```

TYPO3_4-6

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   1.461 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    68.44 [#/sec] (mean)
Time per request:       14.612 [ms] (mean)
Time per request:       14.612 [ms] (mean, across all concurrent requests)
Transfer rate:          82.74 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:    12   14   1.7     15      26
Waiting:       12   14   1.7     14      26
Total:         13   15   1.7     15      26

Percentage of the requests served within a certain time (ms)
  50%     15
  66%     15
  75%     15
  80%     15
  90%     16
  95%     16
  98%     20
  99%     26
 100%     26 (longest request)
```

TYPO3_4-7

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done
```

```
Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   1.388 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    72.02 [#/sec] (mean)
Time per request:       13.884 [ms] (mean)
Time per request:       13.884 [ms] (mean, across all concurrent requests)
Transfer rate:          87.08 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:    12   14   2.1     13      28
Waiting:       12   13   2.1     13      28
Total:         12   14   2.1     13      29

Percentage of the requests served within a certain time (ms)
  50%     13
  66%     14
  75%     14
  80%     14
  90%     15
  95%     18
  98%     22
  99%     29
 100%     29 (longest request)
```

TYPO3_6-0

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   6.062 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    16.50 [#/sec] (mean)
Time per request:       60.620 [ms] (mean)
Time per request:       60.620 [ms] (mean, across all concurrent requests)
Transfer rate:          19.94 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:    57   60   4.2     61      94
Waiting:       57   60   4.2     61      94
Total:         57   61   4.3     61      94

Percentage of the requests served within a certain time (ms)
  50%     61
```

```
  66%     61
  75%     62
  80%     62
  90%     63
  95%     65
  98%     70
  99%     94
 100%     94 (longest request)
```

TYPO3_6-1

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   11.001 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    9.09 [#/sec] (mean)
Time per request:       110.012 [ms] (mean)
Time per request:       110.012 [ms] (mean, across all concurrent requests)
Transfer rate:          10.99 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:    90  110   8.1    109     137
Waiting:       90  109   7.7    109     136
Total:         90  110   8.2    109     137

Percentage of the requests served within a certain time (ms)
  50%    109
  66%    112
  75%    112
  80%    113
  90%    121
  95%    124
  98%    135
  99%    137
 100%    137 (longest request)
```

master as of today

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   14.083 seconds
```

```
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    7.10 [#/sec] (mean)
Time per request:       140.831 [ms] (mean)
Time per request:       140.831 [ms] (mean, across all concurrent requests)
Transfer rate:          8.58 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:   126  141   7.6    139     188
Waiting:      126  140   7.6    139     188
Total:        126  141   7.7    139     190

Percentage of the requests served within a certain time (ms)
  50%    139
  66%    141
  75%    143
  80%    144
  90%    147
  95%    155
  98%    165
  99%    190
 100%    190 (longest request)
```

#6 - 2013-10-20 17:37 - Markus Klein

Do I see that right that this means:

Longest request 4.5 vs 6.0: factor ~4 slower
Longest request 6.0 vs 6.2: factor ~2 slower
Longest request 4.5 vs 6.2: factor ~8 slower

Oh darling....

Did you use PHP 5.3 as specified in the ticket here?

#7 - 2013-10-20 19:15 - Philipp Gampe

Yes I am using **Ubuntu LTS 12.04** which ships **PHP 5.3.10** with Suhosin. I expect that PHP 5.4 is 20-30% faster. PHP 5.5 is even faster.
Keep in mind that this is a VM and that I/O increase count higher here (more PHP cache files).

Also this is a very simple site. Complex sites might do better. Although I did a full cache scenario, therefore this are important numbers nevertheless.

```
<!DOCTYPE html>
<html lang="en">
<head>

<meta charset="utf-8">
<!--
    This website is powered by TYPO3 - inspiring people to share!
    TYPO3 is a free open source Content Management Framework initially created by Kasper Skaarhoj and licensed
 under GNU/GPL.
    TYPO3 is copyright 1998-2013 of Kasper Skaarhoj. Extensions are copyright of their respective owners.
    Information and contribution at http://typo3.org/
-->

<title>Root-Simple-TEXT</title>
<meta name="generator" content="TYPO3 6.2 CMS">

<link rel="stylesheet" type="text/css" href="typo3temp/stylesheet_8b9c2e8d89.css?1382018318" media="all">

</head>
<body>
<h1>Headline</h1><p>Some longer Text
which might be split across several lines
</p><div>Some footer</div>

</body>
</html>
```

This are just three TEXT objects.

#8 - 2013-10-20 19:24 - Markus Klein

Yes, especially when the page is extremely simple it shows that the surrounding overhead got slower a lot.

#9 - 2013-10-20 20:46 - Philipp Gampe

I do not think that we can justify an 400% increase for fully cached pages, because this is the most important scenario.

We really need to find out where this comes from and if we can take certain shortcuts on fully cached pages.

#10 - 2013-10-21 08:25 - Xavier Perseguers

PHP 5.4 is IMHO a must for TYPO3 6.x. This way we already "regain" a bit (although 4.5 would be equally faster as well). Fully cached pages are for sure extremely important and to be optimized first but we should not forget that in many cases USER_INT have to be found. Just think about the felogin which is not AJAX capable.

#11 - 2013-10-21 09:13 - Alexander Jahn

Philipp,

would you mind installing APC and using it instead of *t3lib_cache_backend_FileBackend*? (see http://wiki.typo3.org/Caching_framework#t3lib_cache_backend_ApcBackend)

```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['caching']['cacheConfigurations']['extbase_reflection']['backend']     = 't3lib_cache_backend_ApcBackend';
$GLOBALS['TYPO3_CONF_VARS']['SYS']['caching']['cacheConfigurations']['extbase_object']['backend']         = 't3lib_cache_backend_ApcBackend';
```

We found this has a huge impact on performance.

This should especially be true on an IO-limited VM.

#12 - 2013-10-21 11:34 - Frank Gerards

@Xavier: I see this as you do, but in fact about 75% of the (german) hosting companies dont even offer PHP 5.4 (not to think of PHP 5.5), so TYPO3 6.x MUST run
with maximum performance on PHP 5.3...

#13 - 2013-10-21 13:46 - Philipp Gampe

@Alexander I am aware of APC and the caching framework, however you (usually) do not a memory cache on shared hostings.

@Xavier the impact of PHP 5.4 and 5.5 is much higher on 6.x, because we make better use of OOP then in old CMS installations.

@Frank choosing the right hoster should not be that difficult. You can get good hosting for 10-15€/month (120-180€/a). That should be affortable for every buisiness, especially if their website is not just a place to post the contact information.

What I want to archive with this issue is:

- find the bottlenecks introduced with the latest versions
- (if possible) fix the code or
- find shortcuts for fully cached pages (parameters in matching chash)

#14 - 2013-10-21 13:46 - Philipp Gampe

- Status changed from New to Accepted

setting to accepted as the problem can be easily reproduced

#15 - 2013-10-21 14:31 - Xavier Perseguers

Intro package, without cache

- Open homepage of intro package in 6.2, after clearing all caches
- main() takes 4,235,874 µs to complete
  - run_init::tslib/index_ts.php takes 4,215,982 µs
    - TYPO3\CMS\Core\Core\Bootstrap::loadConfigurationAndInitialize takes 3,108,059 µs
      - TYPO3\CMS\Core\Core\Bootstrap::initializePackageManagement takes 3,100,639 µs
        - TYPO3\CMS\Core\Package\PackageManager::initialize takes 3,092,816 µs
          - TYPO3\CMS\Core\Core\ClassLoader::setPackages takes 3,084,159 µs
            - TYPO3\CMS\Core\Core\ClassAliasMap::setPackages takes 3,083,876 µs
              - TYPO3\CMS\Core\Core\ClassAliasMap::buildMappingFiles takes 3,034,559 µs

Intro package, second run

- main() takes 746,521 µs to complete
  - run_init::tslib/index_ts.php takes 730,117 µs
    - 1. TYPO3\CMS\Frontend\Page\PageGenerator::renderContent takes 340,651 µs
    - 2. TYPO3\CMS\Core\Core\Bootstrap::loadExtensionTables takes 102,665 µs

So without cache, I would start by optimizing PackageManager

#16 - 2013-10-21 15:19 - Thorsten Kahler

Xavier Perseguers wrote:

> [...]
>
> So without cache, I would start by optimizing PackageManager

+1

I'seen comparable numbers on my dev VM and also typo3-6-2beta1-fully-cached2.png points in that direction (24.33 % in Core\Cache\Frontend\PhpFrontEnd::requireOnce()).

#17 - 2013-10-21 15:30 - Thomas Maroschik

Keep in mind that the cache for the Package Manager and Class Loader are built completely upfront to prevent keeping the alias and class map in memory. These arrays contain about 2500 entries each on average and 3 of them have to be kept in memory.
I'm currently exploring a different approach to this, so please consider optimizing the queries of the session management for frontend and backend users for now which send about 2 queries each for example. Usually noone runs TYPO3 without caches in production and the first single request will take some seconds untill all caches are populated. So I would propose to optimize the cached requests first, and with cached I mean USER and USER_INT as the latter is also cached somehow.

#18 - 2013-10-21 17:20 - Philipp Gampe

I fully second Tom here. PackageManager Caches are filled on first request and if you do not change packages (which happens rarely on life sites), then the package manager cache will always be "warm".

I also stumbled over session management, which seems to be really ineffective. Having a backend user cookie (not taken into account above as request came from ab) will add at least another three milliseconds even if you are not logged in.

#19 - 2013-10-21 17:39 - Alexander Opitz

Thorsten Kahler Kahler

If you do profiling with Xdebug or something else, you will ever see such things like "24.33 % in Core\Cache\Frontend\PhpFrontEnd::requireOnce()" for files that will be included. To get more real numbers you should change your setup.

For Xdebug, disable xdebug.collect_includes (http://xdebug.org/docs/all_settings) to get better numbers about the parts that do the trouble.

#20 - 2013-10-21 18:17 - Philipp Gampe

No this is without xdebug, but with xcache (32MB) enabled. All posted numbers are the second or third run, thus the xcache was always warm.

#21 - 2013-10-21 18:20 - Ingo Schmitt

Are these numbers generated with a bytecode cache or without?

#22 - 2013-10-21 18:23 - Philipp Gampe

Interesting: Without xcache, the performance seems to be better.

(I will provide the numbers without bytecode cache in the next hour; **the next two blocks are generated while running on battery, thus might be lower the needed**).

TYPO3_4-5 slower without bytecode cache by a factor 4.5

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
```

```
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        937 bytes

Concurrency Level:      1
Time taken for tests:   5.310 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      123800 bytes
HTML transferred:       93700 bytes
Requests per second:    18.83 [#/sec] (mean)
Time per request:       53.099 [ms] (mean)
Time per request:       53.099 [ms] (mean, across all concurrent requests)
Transfer rate:          22.77 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:    48   53   4.4     52      81
Waiting:       47   53   4.4     52      81
Total:         48   53   4.5     52      82

Percentage of the requests served within a certain time (ms)
  50%     52
  66%     53
  75%     56
  80%     57
  90%     58
  95%     59
  98%     61
  99%     82
 100%     82 (longest request)
```

master faster without bytecode cache ???

```
$ ab -n 100 "http://ubuntu-typo3.local/dummy/index.php?id=1"
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ubuntu-typo3.local (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        ubuntu-typo3.local
Server Port:            80

Document Path:          /dummy/index.php?id=1
Document Length:        752 bytes

Concurrency Level:      1
Time taken for tests:   9.905 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      105300 bytes
HTML transferred:       75200 bytes
Requests per second:    10.10 [#/sec] (mean)
Time per request:       99.052 [ms] (mean)
Time per request:       99.052 [ms] (mean, across all concurrent requests)
Transfer rate:          10.38 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:    85   99   5.6     99     134
Waiting:       85   99   5.6     99     134
Total:         86   99   5.7    100     135

Percentage of the requests served within a certain time (ms)
  50%    100
  66%    101
```

```
 75%    101
 80%    102
 90%    103
 95%    105
 98%    114
 99%    135
100%    135 (longest request)
```

#23 - 2013-10-21 18:28 - Alexander Opitz

@Philipp:

> No this is without xdebug, but with xcache (32MB) enabled. All posted numbers are the second or third run, thus the xcache was always warm.<<<

I spoke to Thorsten Kahler and not to you and your numbers. You also do benchmarking and not profiling.

#24 - 2013-10-22 12:32 - Jochen Weiland

Let us know if you want to test this in one of our hosting packages. There you can switch instantly between PHP 5.3/5.4 and enable/disable APC, i.e. use the same setup with different PHP/APC configurations.

#25 - 2013-10-22 12:45 - Alexander Opitz

Hi Jochen,

thats a nice offer. We should define scenarios first what we need to benchmark, the single page from cache isn't the only speed problem.

#26 - 2013-10-22 12:50 - Thorsten Kahler

Alexander Opitz wrote:

> @Philipp:
>
> > No this is without xdebug, but with xcache (32MB) enabled. All posted numbers are the second or third run, thus the xcache was always warm.<<<
>
> I spoke to Thorsten Kahler and not to you and your numbers. You also do benchmarking and not profiling.

@Alex: I referred to Philipps diagram, so Philipp is the right one to answer.

#27 - 2013-10-22 13:59 - Dmitry Dulepov

PhpFrontend::requireOnce() does not do anything except:

```
return $this->backend->requireOnce($entryIdentifier);
```

I profiled 6.2 with XDebug, top time eating classes are from the caching framework and class loader.

ygqs.png
Notice the last line in the image. File typo3temp/Cache/Code/cache_classes/tx_realurl.php contains the following:

```
<?php require __DIR__ . '/../../../../typo3conf/ext/realurl/class.tx_realurl.php';
```

What is that? Including the file from cache , which just includes the real file? If it is the same for every other file, we double amount of require_once classes, which **is** slow because it works with the file system.

#28 - 2013-10-22 14:33 - Alexander Opitz

Thorsten Kahler

Oh I thought it was your diagram, shame on me.

But in real the require_once isn't such cost expensive without profiling, I got such high numbers with Xdebug before I disabled "xdebug.collect_includes".

#29 - 2013-10-23 08:16 - Alexander Opitz

Dmitry Dulepov

In refer to your message: "What is that? Including the file from cache , which just includes the real file?"

Please take a look at http://wiki.typo3.org/Blueprints/Packagemanager#Implementation_Details why we have this handling.

#30 - 2013-10-23 11:21 - Alexander Opitz

all review testers

After profiling and writing with Thomas Maroschik, I was able to identify a problem, which may also affect your tests.
If you have a symlink inside the path to your "/typo3temp/Cache/Code/" directory, the caching may fail.

To verify this problem, open in /typo3temp/Cache/Code/ your PackageManager_*.php and remove the following Part

```
__DIR__ !== '{your path}/typo3temp/Cache/Code/cache_core' ? FALSE :
```

so your cache file should be read:

```
array (
  'packageStatesConfiguration' =>
  array (
    'packages' =>
    array (
...
```

And benchmark your installation again.

On my installation the reqests per second changed
with Xdebug from 145 to 229
without Xdebug from 221 to 305

#31 - 2013-10-23 14:21 - Philipp Gampe

Thanks Alexander.

Anyway, my numbers are from a default install with typo3_src link structure and real folders for the rest (automatically created by the installer).

#32 - 2013-10-25 11:02 - Dmitry Dulepov

@Alexander Opitz

Hmmm, this is interesting :)

Do you mean "The backend creates "proxy require" files so that the memory consumption of the autoloader is lowered"? Was memory a real issue for anybody? To me it does not look good to solve memory issues by doubling amount of inclusions.

After I removed that line, I got 6.78 rq/s. With that line it is 3.28 rq/s. Used "ab -n 100" for testing. What hardware do you have with such rq/s numbers?

#33 - 2013-10-25 11:36 - Alexander Opitz

Dmitry Dulepov

About memory consumption and the "proxy require" I can't tell much. The Blueprint is written by Thomas Maroschik. But while developing the PackageManager I did benchmarking the autoloader from TYPO3 6.0/6.1 his biggest issues are memory consumption for all the aliases parts. First try to remove this was eliminating the aliasing if not needed. But we came to the point where an alias is needed but isn't autoloaded. The second try from Thomas was using a Symlink structure. It was faster but APC and Xcache didn't performed well on them as they had problems with caching the data. So Thomas came with this solution of the "proxy require", they will be cached by APC, Xcache or OPcache. While using Xdebug, xhprof or something like that it will decrease the performance more then the symlink solution (thats what I get from my numbers).

The mentioned line doesn't have to do with this "proxy require" it is more or less a nitpik which may be removed, it prevents using the PackageManager and PackageObject cache which do not have to do with the "proxy require".

My Hardware/Software:

- 8 core 64Bit CPU
- 12 GB Ram
- Linux Kernel 3.11
- apache 2.4.6
- PHP 5.5.3 with OPcache 7.0.3
- MySQL 5.5.34

Test with "ab -n 1000 -c 100 typo3-master.sphinx"

#34 - 2013-11-04 20:29 - Mathias Schreiber

Is it possible to get some cachegrind dumps?
I am rather busy this week, but reading a cachegrind file might be possible.

Just don't have the time to set up my whole profiling env this week

#35 - 2013-11-04 22:06 - Philipp Gampe

@Mathias please have a look into the profiles.tar.gz attached to the issue. It includes some "callgrind" files for both 4.5 and 6.2; keep in mind that they are create with a different tool the usual cachegrind files, but work fine in e.g. kcachegrid.

#36 - 2013-11-04 22:44 - Mathias Schreiber

Thanks.
I need to check the includes and called functions to determine the version or do you have it in mind which dump was what?

Apart from that:
What do you think about search & replace-no-brainer optimizations?
Things that come to mind:

- is_file vs. file_exists
- isset vs array_key_exists
  Please also keep in mind that I didn't dig into the code yet.
  When you cache file paths you check, do you use absolute paths?
  If not, let's do that as well, since it will remove the entire include_path checks as well.

From what I've seen so far the whole "is the file there" things takes up a huge chunk of time.
This would be a good thing to start digging.

Let's have a chat on that this weekend

#37 - 2013-11-04 23:11 - Markus Klein

@Mathias:
Is is_file really so much faster in the end? Both need to stat the file by the OS syscall.

isset is indeed faster, but it behaves differently when it comes to array values being NULL.
See example#2 here http://us2.php.net/array_key_exists

Therefore such changes might be dangerous.

#38 - 2013-11-05 08:28 - Mathias Schreiber

Markus Klein wrote:

> @Mathias:
> Is is_file really so much faster in the end? Both need to stat the file by the OS syscall.

I made 3 benchmarks on this matter.

- SAN FS (Nexenta) mounted via NFS (Ubuntu 12.04): 12% increase
- SAN FS mounted via iSCSI onto VMWare ESXI (Ubuntu 12.04): 48% increase
- Samsung 840 EVO Native OSX Mountain Lion: 120% increase

So no, it is not "so much faster", but going 1kph faster still makes you win the race :)
I suggest to improve whereever we can, even for the slightest improvement.
I read something interesting about not using require but using include instead, since most of the time you try/catch that anyways (or exception out of it at least).
So it might be possible to get rid of a lot the file_exists (don't think all will be possible, though).

> isset is indeed faster, but it behaves differently when it comes to array values being NULL.
> See example#2 here http://us2.php.net/array_key_exists

Yeah, that's a typical one.
Personally I consider null values in arrays bad practice but opinions may of course differ here.
At least we should try to get rid of these whereever possible.
Maybe even change them all and then check what breaks.
The speed improvement is tremendous.

> Therefore such changes might be dangerous.

No pain, no gain.
If the system was like 10% slower, fine - I wouldn't even bother.

But 800% or heck... even 50% is something we just cannot allow ourselves.

#39 - 2013-11-05 08:42 - Xavier Perseguers

I agree that we **must** do whatever we can to improve performance. However I would prefer to profile typical calls, find 1-2 main bottlenecks, fix them whichever way makes it drop dramatically in term of cost and start again.

#40 - 2013-11-05 08:44 - Alexander Opitz

I think we also can't change all file_exists into is_file, as is_file returns false for symlinks.

@Mathias: Which PHP version did you use for your benchmarks?

#41 - 2013-11-05 09:06 - Mathias Schreiber

Alexander Opitz wrote:

> I think we also can't change all file_exists into is_file, as is_file returns false for symlinks.

Well, don't get me wrong, I don't know the "new" core and don't have the time to take a look at it since I have a company to run.
I can just take a look at cachegrinds and similar and come up with proposals.
I have to leave it up to you to decide which improvement belongs at which point in the code.

> @Mathias: Which PHP version did you use for your benchmarks?

I used 5.3.25 or 27.
Just to suppress all "with 5.x everything will be faster/better/bla" that might come up (ignore if you don't feel addressed :)):
We can't afford to wait for stuff like that.
5.3 is the major player out in the field, waiting for newer versions just states being lazy.
If newer PHP version speed things up - even better.
But as for now we need to focus on what the market demands and the competition delivers in order to stay competitive. And just in case this didn't come through... that's the only thing I care about - TYPO3 being competitive.
If you guys get that done with the "100.000 files to require, although logic dictates it is slower" approach, that's fine with me.
We all just need to be on the same track here.
So sacrifying competitiveness for "by-the-book-code" is not an option :)

Xavier Perseguers wrote:

> find 1-2 main bottlenecks, fix them whichever way makes it drop dramatically in term of cost and start again.

agreed.
Apart from that I will try to convince Rupi to revive and supply his old testing setup so we all have a common base to test on.
Luckily this instance covers what you mean by "typical calls", since it features basically everything a 4.3 frontend offered in that time.

So all in all:
I love everything you do for TYPO3, as long as you love everything I do for TYPO3 - joined forces for the greater good.

#42 - 2013-11-05 19:22 - Dmitry Dulepov

Probably we can set something to cache at APC instead of file system if APC is available.

#43 - 2013-11-05 22:40 - Steffen Müller

Dmitry Dulepov wrote:

> Probably we can set something to cache at APC instead of file system if APC is available.

Is APC meanwhile save on shared hosting environments to enable it by default on any installation? That was a blocker some years ago. Is this still true?

#44 - 2013-11-05 22:54 - Ingo Schmitt

APC ist still a problem in shared hosting when you not use FCGI. A solution more general would be better! And every solution which ist fast without any extra software would be even faster with a bytecode cache!

#45 - 2013-11-06 10:59 - Ernesto Baschny

The Install Tool has a "Configuration Preset" which detects if APC is present and has enough free memory and recommends in this case to switch some Extbase caches to it. So it's not "enabled by default" but a recommended setting through the Install Tool if available (and auto-configured this way if you install through the Step Installer and it detects this situation).

The "default" has still to be File or Database based caching - as these are always available.

#46 - 2013-11-07 11:00 - Alexander Opitz

A speed improvement for fully cached pages: http://forge.typo3.org/issues/53404

#47 - 2013-11-14 23:18 - Markus Klein

> http://www.php.net/ChangeLog-5.php#5.5.6
>
> - Improved performance of array_merge() and func_get_args() by eliminating useless copying.

This could bring quite a bit of performance for TYPO3 since array_merge() is used ~400 times in Core.

(Disclaimer: This is just a note and not an excuse for the necessary software improvements.)

#48 - 2013-11-16 19:49 - Markus Klein

A very small performance thing, yet done on every request: http://forge.typo3.org/issues/53702

#49 - 2013-11-25 23:08 - Alexander Stehlik

I think i found another one:

http://forge.typo3.org/issues/53962

#50 - 2013-12-19 16:23 - Michiel Roos

And this one is a big slowdown: http://forge.typo3.org/issues/53862

#51 - 2014-01-15 14:36 - Alexander Opitz

Since 6.2alpha2, many speedups are implemented. The backend really seams faster now. But on my frontend test (empty cached page), nothing changed (compared alpha2 with master of today). The speedup from #53598 is measurable, but we regressed somewhere else so that we don't see any change in the comparison. Till yet, we are slower then 6.1.7 (for this test case).

How does it look for your tests?
Dmitry Dulepov can you post latest results from your system?

#52 - 2014-01-15 15:19 - Alexander Opitz

Dmitry Dulepov You may need the patch from #55022 for your setup (as described above on comment 30).

#53 - 2014-01-16 16:41 - Ingo Schmitt

- Parent task set to #55065

#54 - 2014-01-20 16:48 - Ingo Schmitt

- Category set to Performance

#55 - 2014-06-19 17:46 - Oliver Hader

- Status changed from Accepted to Needs Feedback

Quite a lot of actions happened in the last few weeks of 6.2 development concerning performance. Personally I consider this issue to be solved. Please give some detailed feedback on more current performance issues, otherwise this issue will be closed. Thanks in advance!

#56 - 2014-06-19 18:00 - Markus Klein

You consider this solved? What about the huge amount of open sub-tickets?

#57 - 2014-06-19 19:01 - Simon Schaufelberger

I would still consider the performance of the new extension manager as horrible, especially on windows. Don't know much about the rest.

#58 - 2015-04-09 17:28 - Markus Klein

- Is Regression changed from Yes to No

#59 - 2015-04-09 17:40 - Kay Strobach

Markus, can you please check the performance 7.x vs. 4.5 should be a lot faster, as we can skip the doubled class loading there ...

#60 - 2015-04-09 17:46 - Markus Klein

I'm not checking anything here, just cleaning up the bug tracker.

#61 - 2015-07-16 02:09 - Benni Mack

- Status changed from Needs Feedback to Resolved

I consider this as resolved. Especially with 6.2.14 and more especially with CMS7.

#62 - 2018-10-02 12:09 - Benni Mack

- Status changed from Resolved to Closed

Files

profiles.tar.gz	315 KB	2013-10-18	Philipp Gampe
typo3-4-5-fully-cached.png	471 KB	2013-10-18	Philipp Gampe
typo3-6-2beta1-fully-cached2.png	1.39 MB	2013-10-18	Philipp Gampe