SAHANA Profiling

Importance of profiling

If it is required to carry out a code optimization task, the developer should know about performance bottlenecks within the system. Otherwise the optimization process would be not effective. if we take the SAHANA system there are so many functions execute to deliver an output. To optimize the SAHANA system we should identify expensive functions those are worth optimizing. To identify expensive functions those may be bottlenecks a profiling exercise should be carried out

To profile the system we use “Xdebug(http://xdebug.org/)” module and “Kcachegrind (http://kcachegrind.sourceforge.net)” visualization tool.

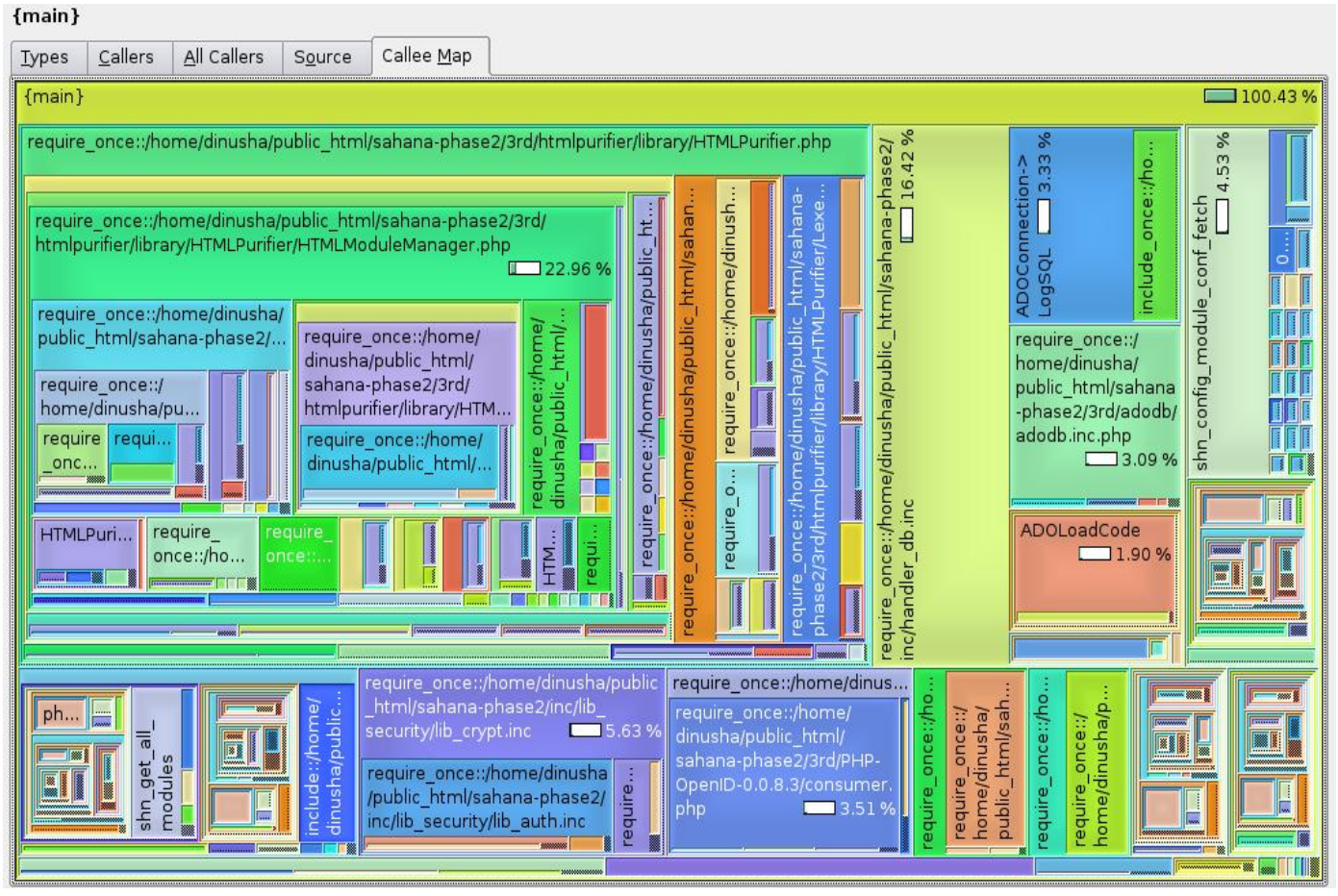
Profiling plan

We start the profiling task with SAHANA index.php file without requesting any other module. by doing that we expect to identify bottlenecks within the SAHANA framework and to present the information flow.

Then we profile each module at a time to identify issues within those modules

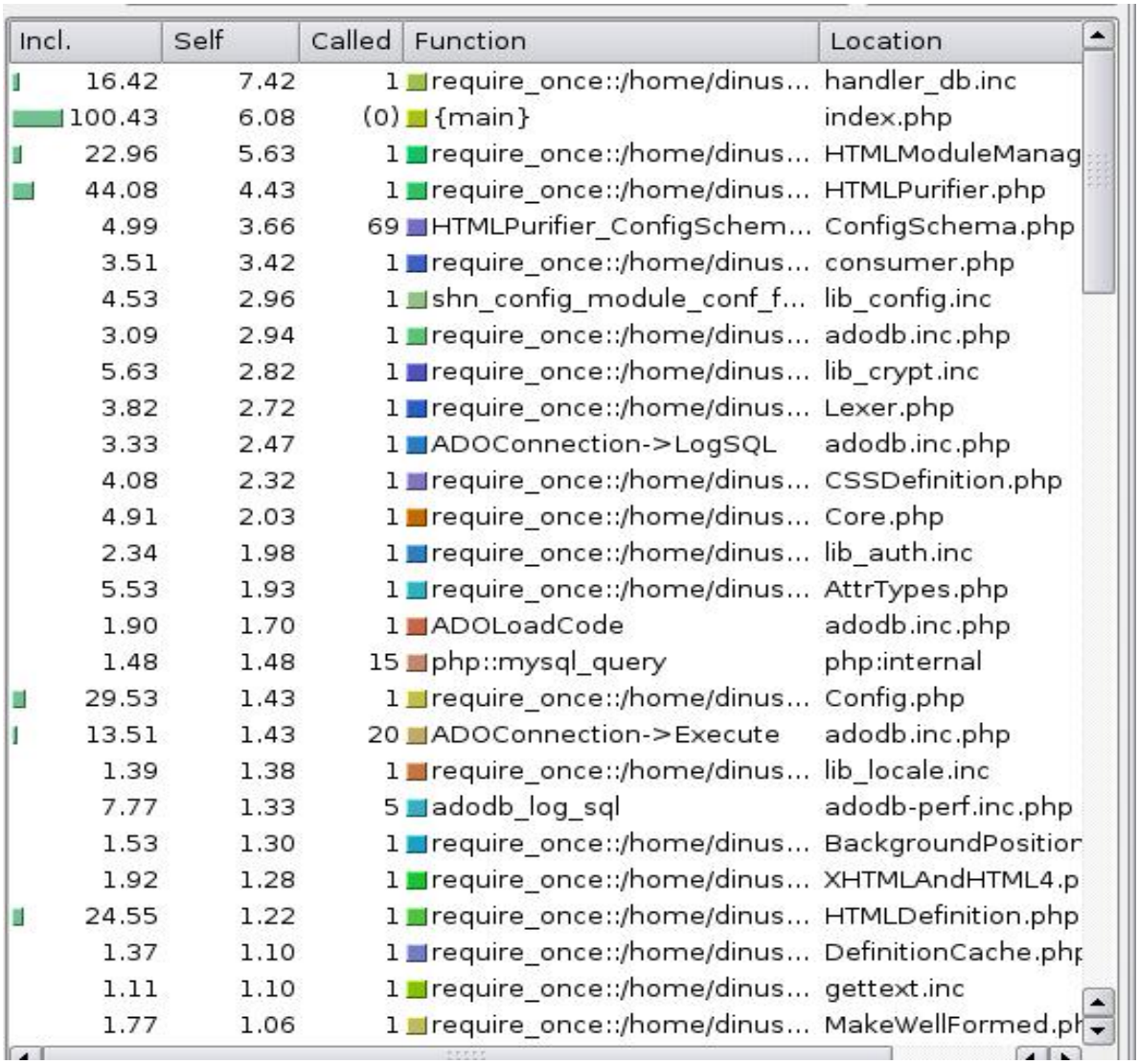
Profiling Index.php

When the index.php file is requested there will be so many functions executed. The following diagram represents all functions.



In the above diagram empty rectangles mean there are no further functions execute within that function. We should filter out important functions those are worth optimizing.

It is important to select most expensive functions for the optimization process to make the overall process effective. The following diagram show most expensive function in terms of memory usage and time consumption.



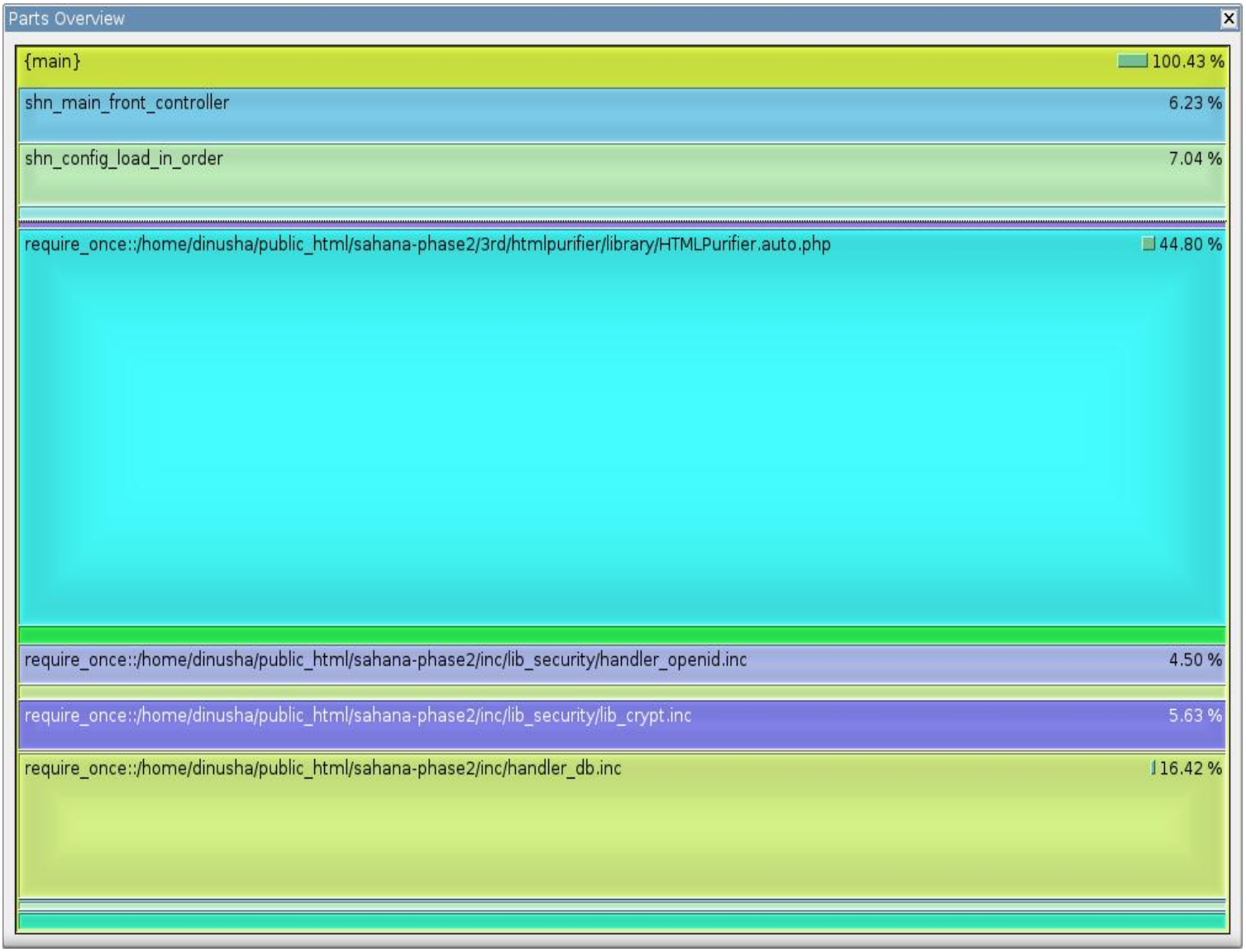
Incl.	Self	Called	Function	Location
16.42	7.42	1	require_once::/home/dinus...	handler_db.inc
100.43	6.08	(0)	{main}	index.php
22.96	5.63	1	require_once::/home/dinus...	HTMLModuleManag
44.08	4.43	1	require_once::/home/dinus...	HTMLPurifier.php
4.99	3.66	69	HTMLPurifier_ConfigSchem...	ConfigSchema.php
3.51	3.42	1	require_once::/home/dinus...	consumer.php
4.53	2.96	1	shn_config_module_conf_f...	lib_config.inc
3.09	2.94	1	require_once::/home/dinus...	adodb.inc.php
5.63	2.82	1	require_once::/home/dinus...	lib_crypt.inc
3.82	2.72	1	require_once::/home/dinus...	Lexer.php
3.33	2.47	1	ADOConnection->LogSQL	adodb.inc.php
4.08	2.32	1	require_once::/home/dinus...	CSSDefinition.php
4.91	2.03	1	require_once::/home/dinus...	Core.php
2.34	1.98	1	require_once::/home/dinus...	lib_auth.inc
5.53	1.93	1	require_once::/home/dinus...	AttrTypes.php
1.90	1.70	1	ADOLoadCode	adodb.inc.php
1.48	1.48	15	php::mysql_query	php:internal
29.53	1.43	1	require_once::/home/dinus...	Config.php
13.51	1.43	20	ADOConnection->Execute	adodb.inc.php
1.39	1.38	1	require_once::/home/dinus...	lib_locale.inc
7.77	1.33	5	adodb_log_sql	adodb-perf.inc.php
1.53	1.30	1	require_once::/home/dinus...	BackgroundPositior
1.92	1.28	1	require_once::/home/dinus...	XHTMLAndHTML4.p
24.55	1.22	1	require_once::/home/dinus...	HTMLDefinition.php
1.37	1.10	1	require_once::/home/dinus...	DefinitionCache.php
1.11	1.10	1	require_once::/home/dinus...	gettext.inc
1.77	1.06	1	require_once::/home/dinus...	MakeWellFormed.ph

In the above diagram the tab “self” indicates the time spent within the respective function. And the tab “incl” represents the time spent within the function with other child functions. So it displays the inclusive cost where as the “self” tab indicates the exclusive cost. The “called” tab represents the number of calls for the respective function.

As an example 16.42% of the cost is incurred within the handler_db.inc including its child functions. Direct functions within that script take 7.42% of the time.

By analyzing the above diagram we can say that most expensive segments for the index.php are within the handler_db.inc, HTMLPurifier, Open-Id and lib_config.inc, etc.

If we further analyze the execution we can determine most expensive parts as displayed in the following image.



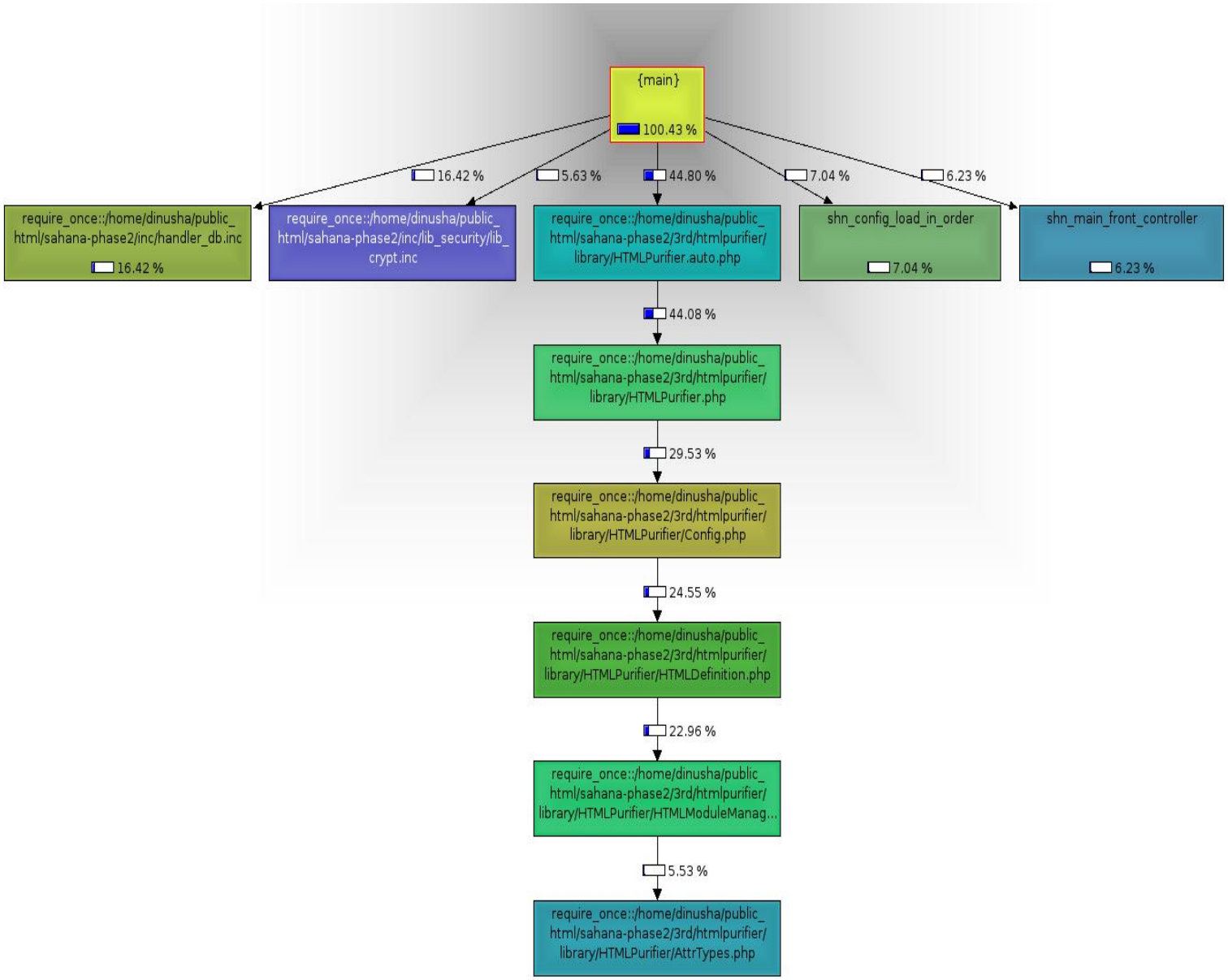
From the total execution time which is 100.43% (it is more than 100% since it shows the inclusive cost)

- HTMLPurifier takes 44.8% ,
- handler_db.inc takes16.42% ,
- shn_config_load_in_order() function takes 7.04% ,
- shn_main_front_controller() function takes 6.23% ,
- lib_crypt.inc takes 5.63% ,
- handler_openid.inc takes 4.5% ,

of the process time.

So it is advisable to concentrate only on those parts when handling the optimization process.

The utilization of the process time can be further explained using the following graph

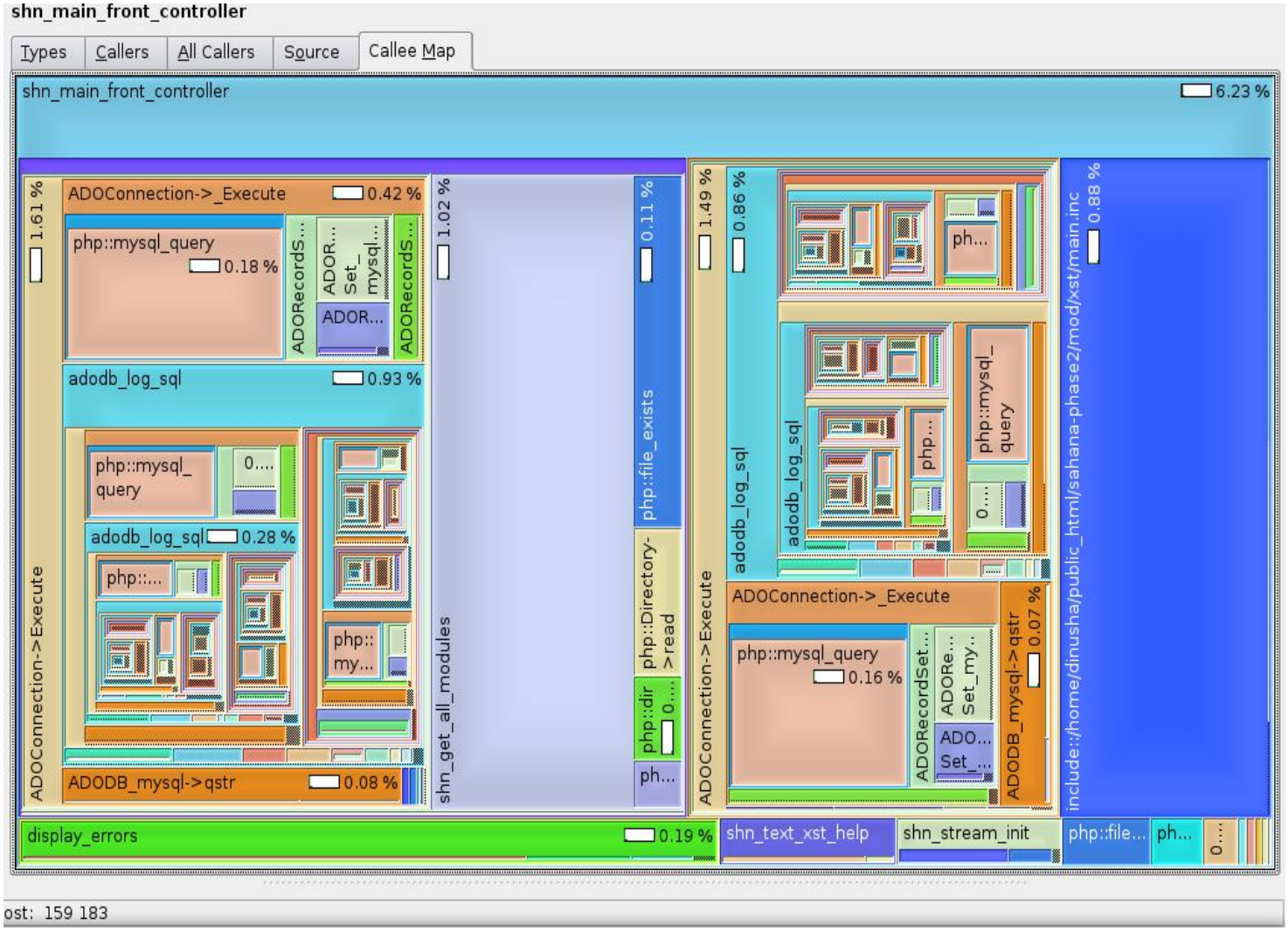


As it is clearly displayed the time is mainly shared among 5 parts listed above. And the HTMLPurifier has some more expensive segments.

The following section would analyze expensive segments thoroughly

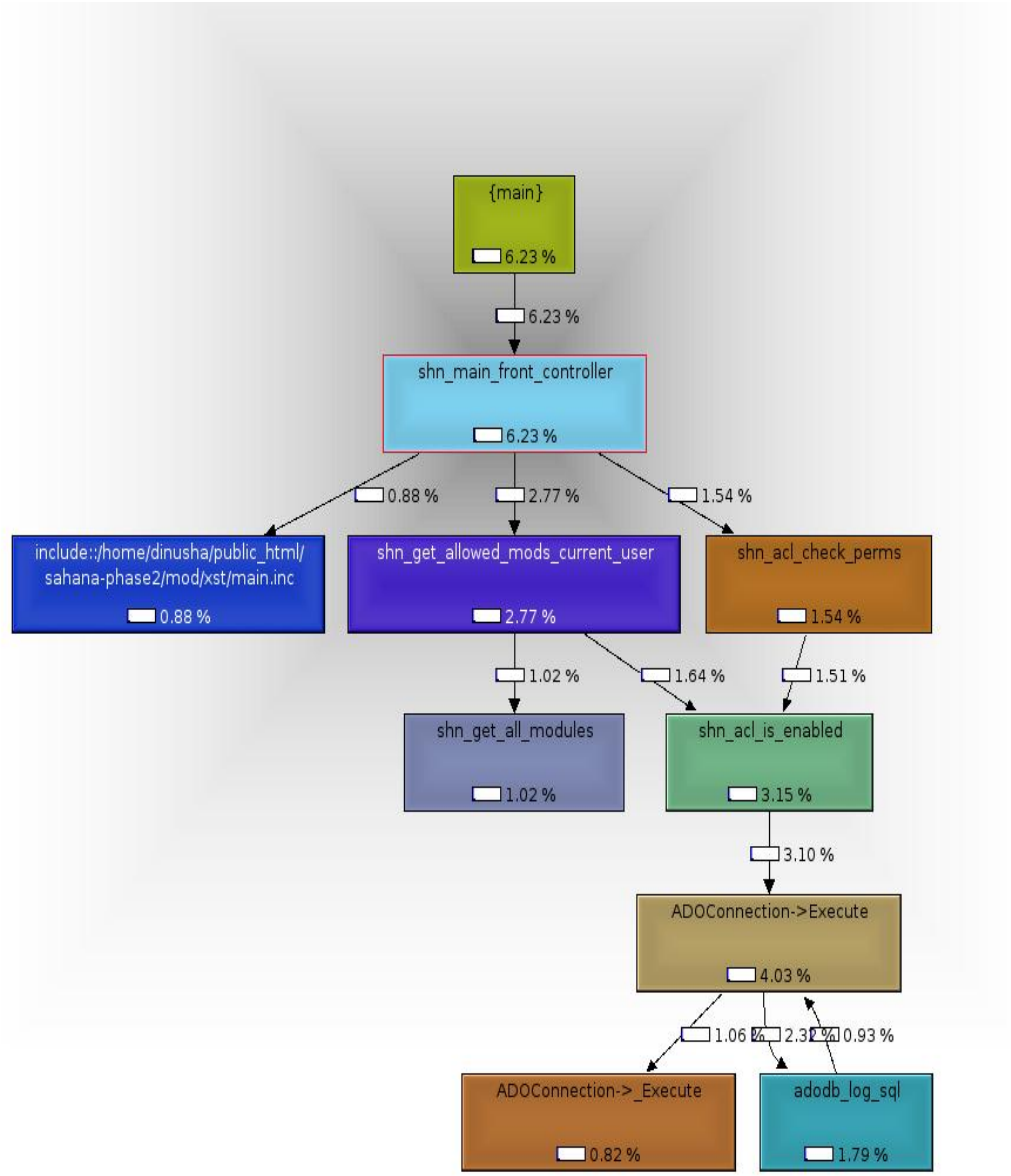
Front Controller

As indicated above the shn_main_front_controller () takes 6.23% of the execution time. All notable functions within that is displayed below



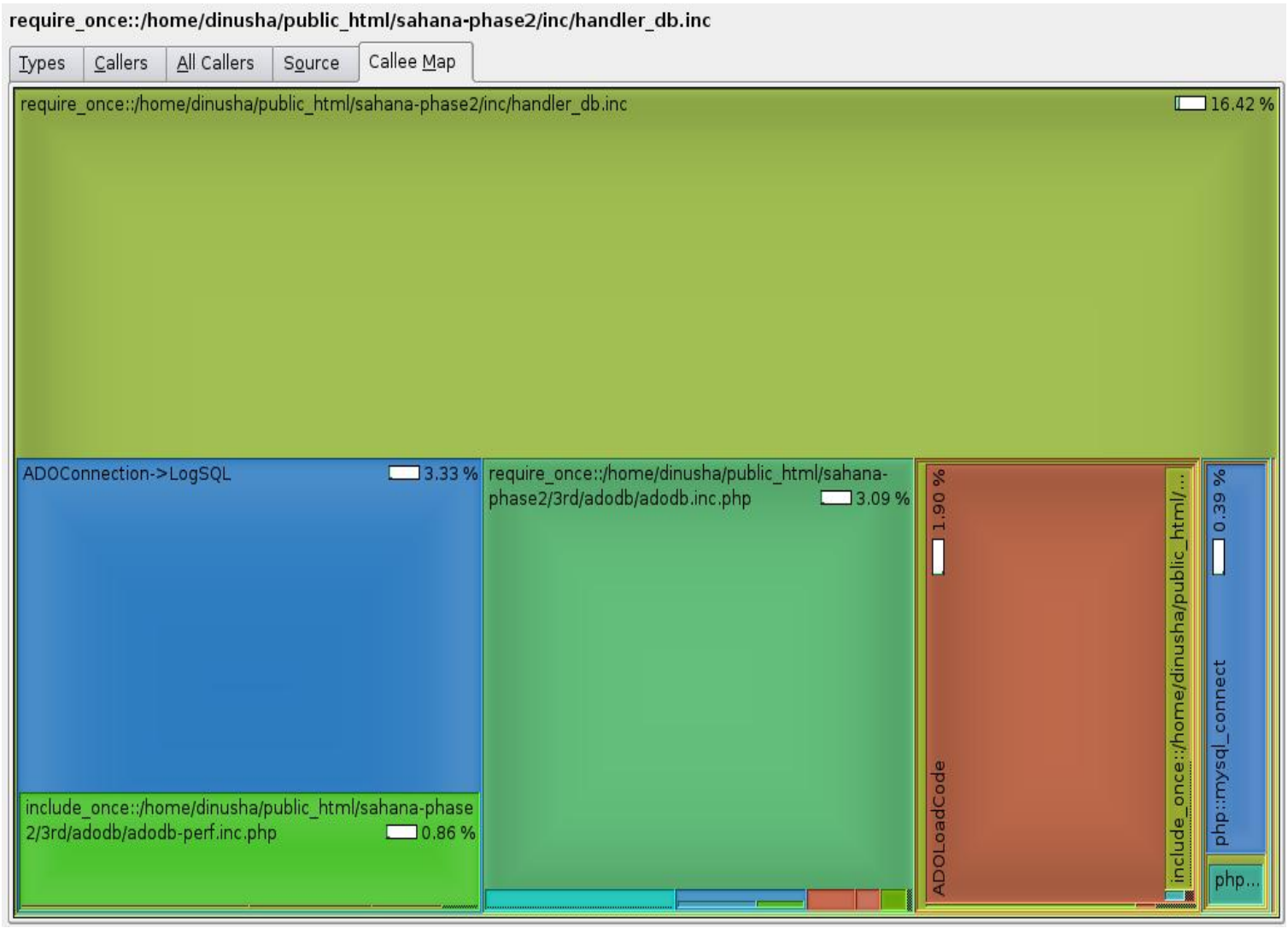
We can see that ADOdb Execute function,shn_get_all_modues() and Functions within XST module are responsible for the execution overhead.

The graph view of the execution process is displayed below

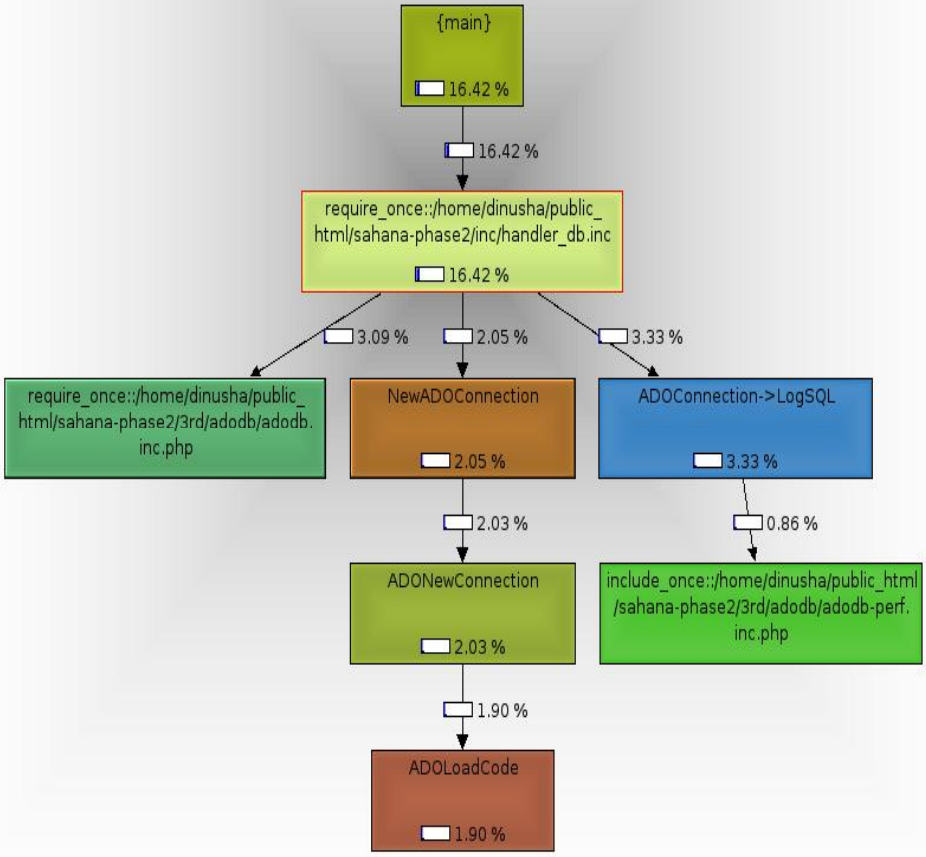


Handler_db

As indicated above functions within handler_db.inc takes 16.42% of the execution time. All notable functions within that is displayed below

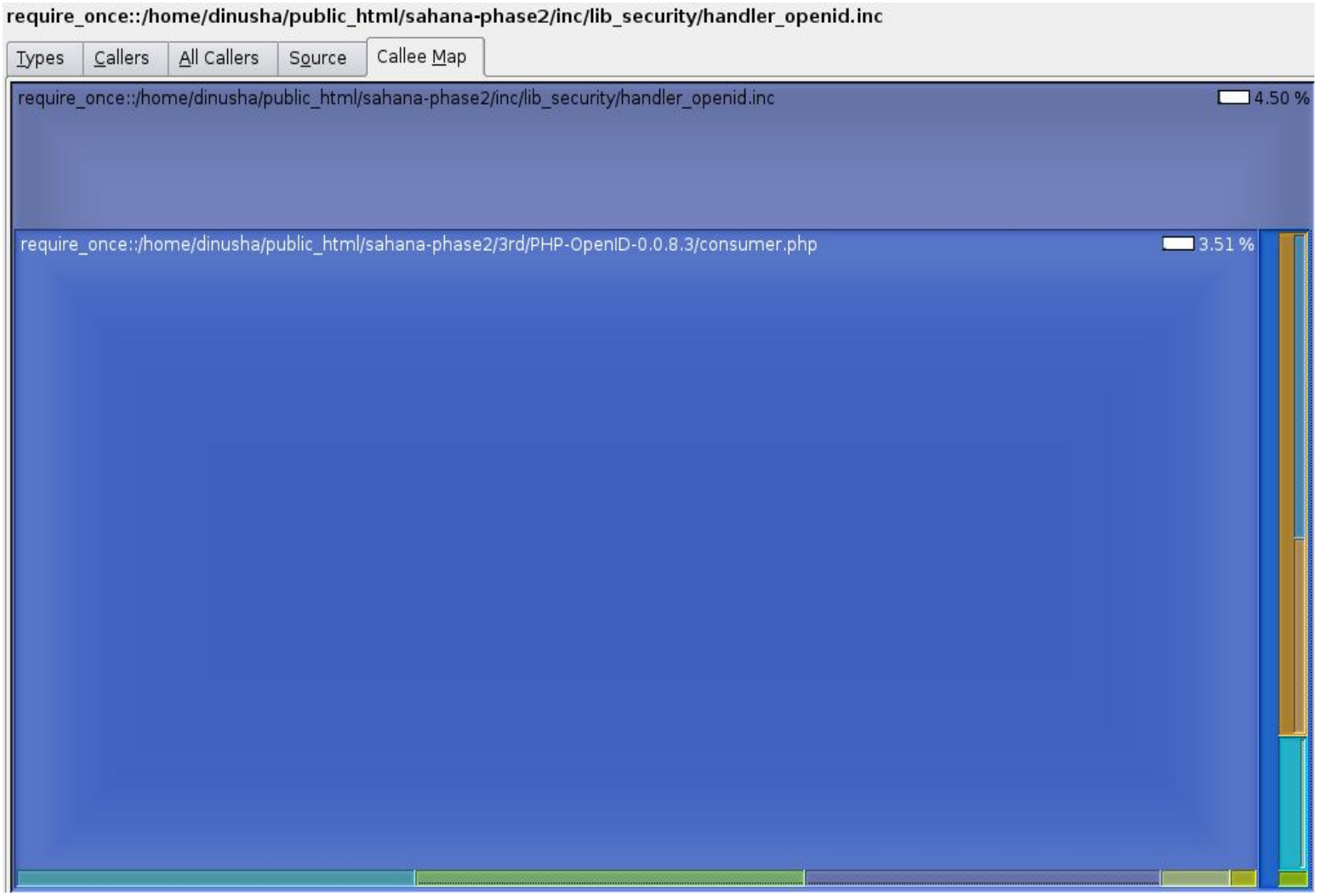


We can see that most of the overhead is spent on ADOdb functions. The graph view of the execution process is displayed below

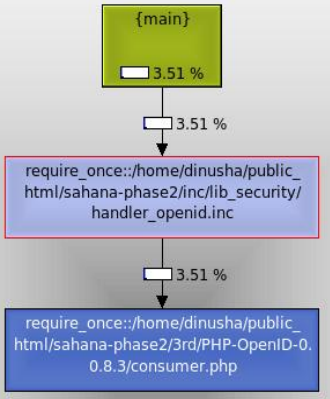

{main}
16.42 %
16.42 %
require_once::/home/dinusha/public_
html/sahana-phase2/inc/handler_db.inc
16.42 %
3.09 %
2.05 %
3.33 %
require_once::/home/dinusha/public_
html/sahana-phase2/3rd/adodb/adodb.
inc.php
NewADOConnection
2.05 %
ADOConnection->LogSQL
3.33 %
2.03 %
0.86 %
ADONewConnection
2.03 %
include_once::/home/dinusha/public_html
/sahana-phase2/3rd/adodb/adodb-perf.
inc.php
1.90 %
ADOLoadCode
1.90 %

Handler_openid

As indicated above functions within handler_openid.inc takes 4.50% of the execution time. All notable functions within that is displayed below

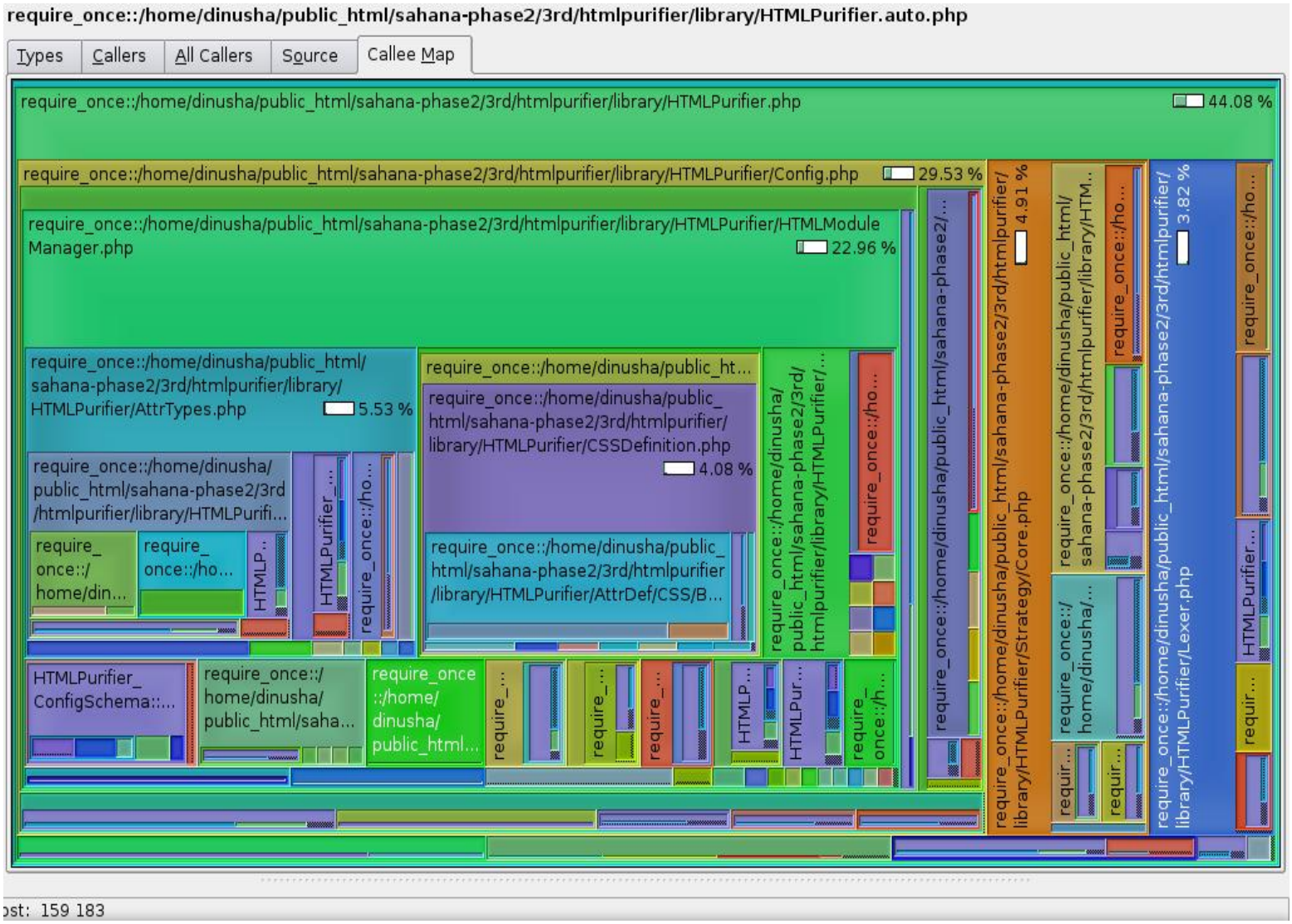


The graph view of the execution process is displayed below

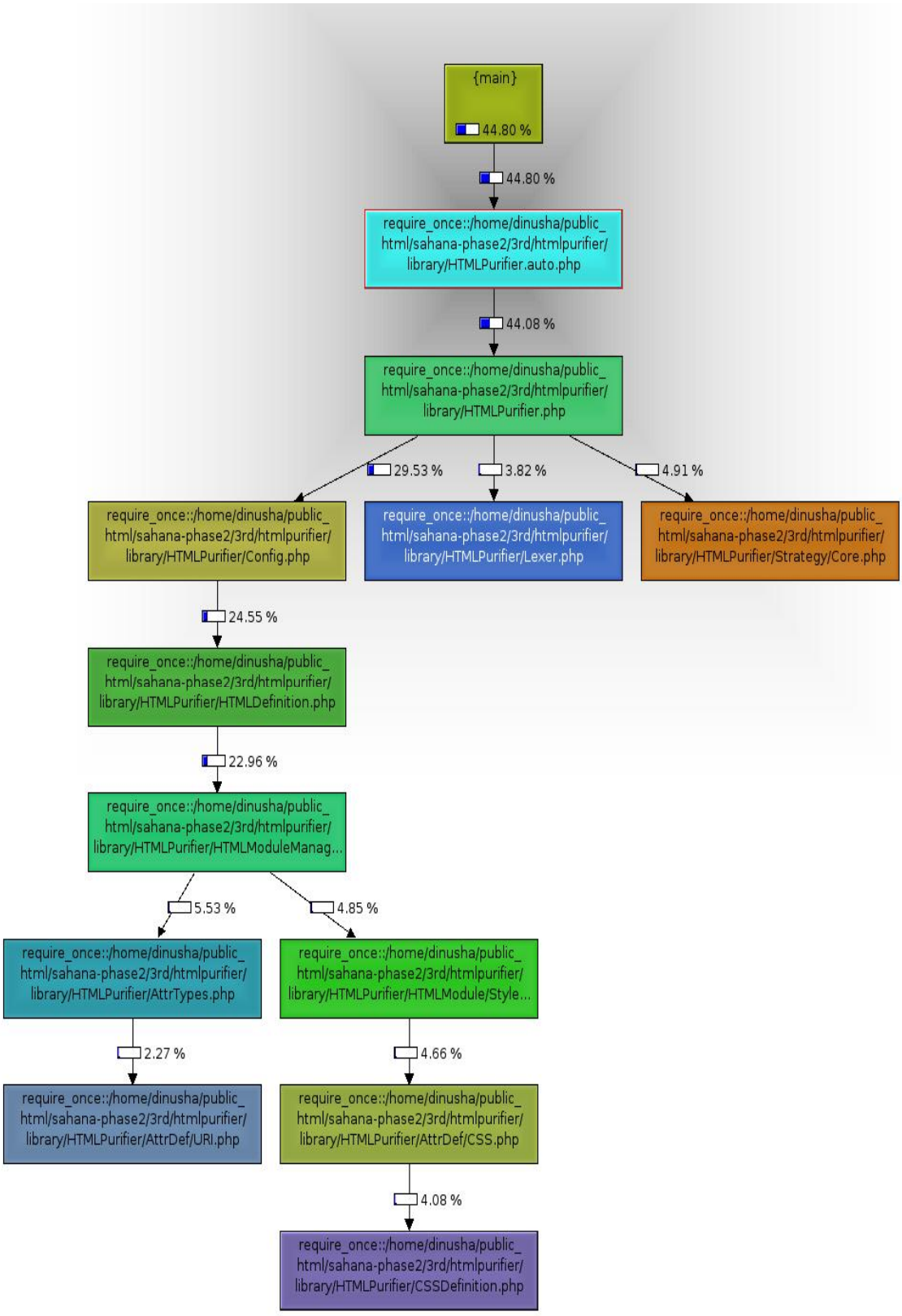


HTMLPurifier

Functions within HTMLPurifier take 44.08% of the execution time. All notable functions within that is displayed below

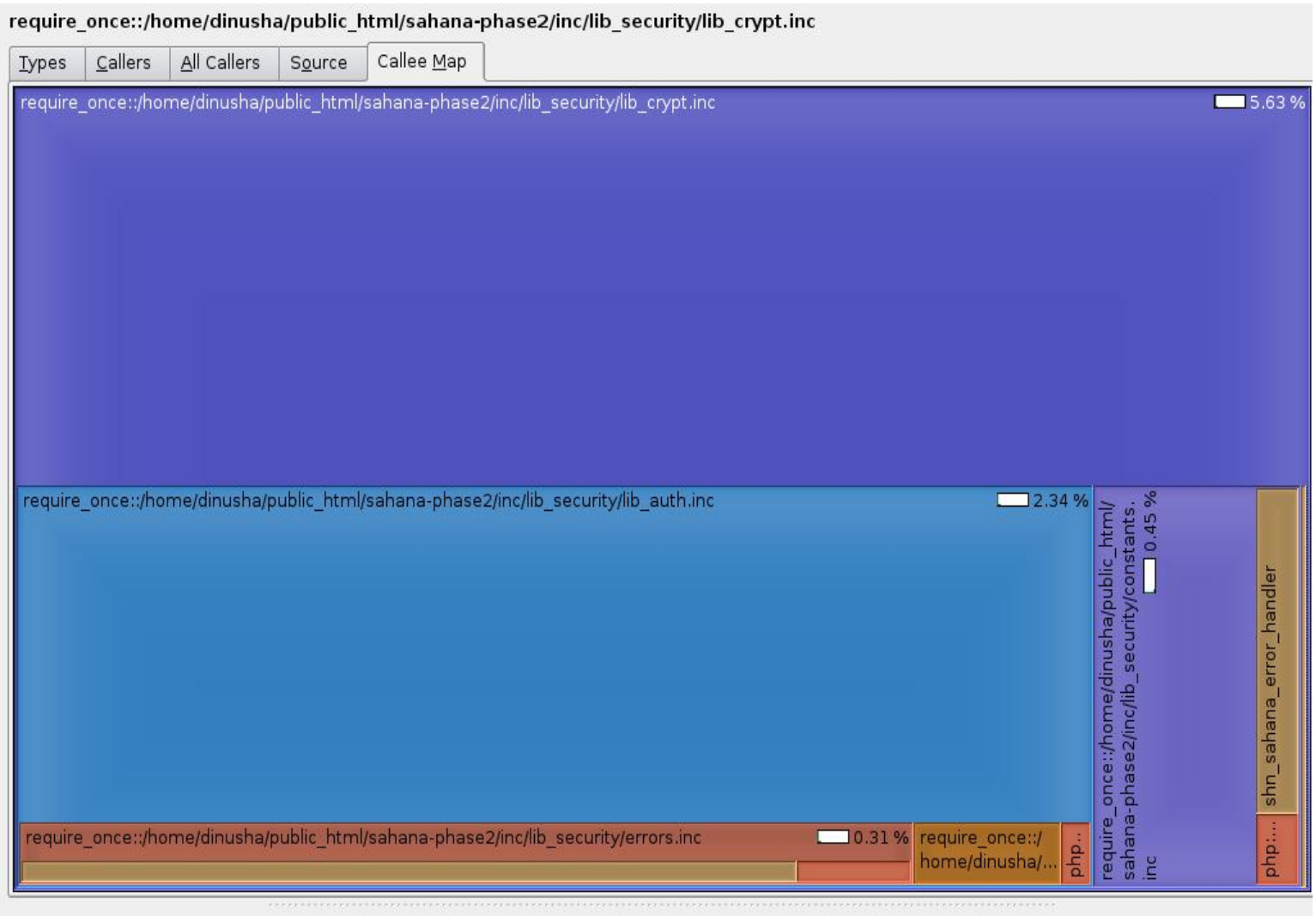


It can be seen that there are so many functions executed within html purifier. We can further analyze using the graph view.

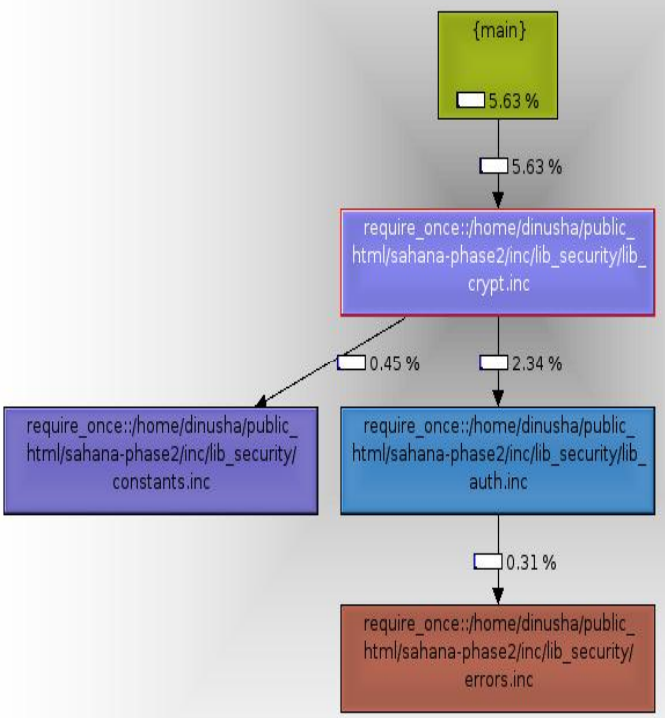

{main}
44.80 %
44.80 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier.auto.php
44.08 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier.php
29.53 %
3.82 %
4.91 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/Config.php
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/Lexer.php
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/Strategy/Core.php
24.55 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/HTMLDefinition.php
22.96 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/HTMLModuleManag...
5.53 %
4.85 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/AttrTypes.php
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/HTMLModule/Style...
2.27 %
4.66 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/AttrDef/URI.php
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/AttrDef/CSS.php
4.08 %
require_once::/home/dinusha/public_html/sahana-phase2/3rd/htmlpurifier/library/HTMLPurifier/CSSDefinition.php

Lib_crypt

Functions within lib_crypt take 5.63% of the execution time. All notable functions within that is displayed below

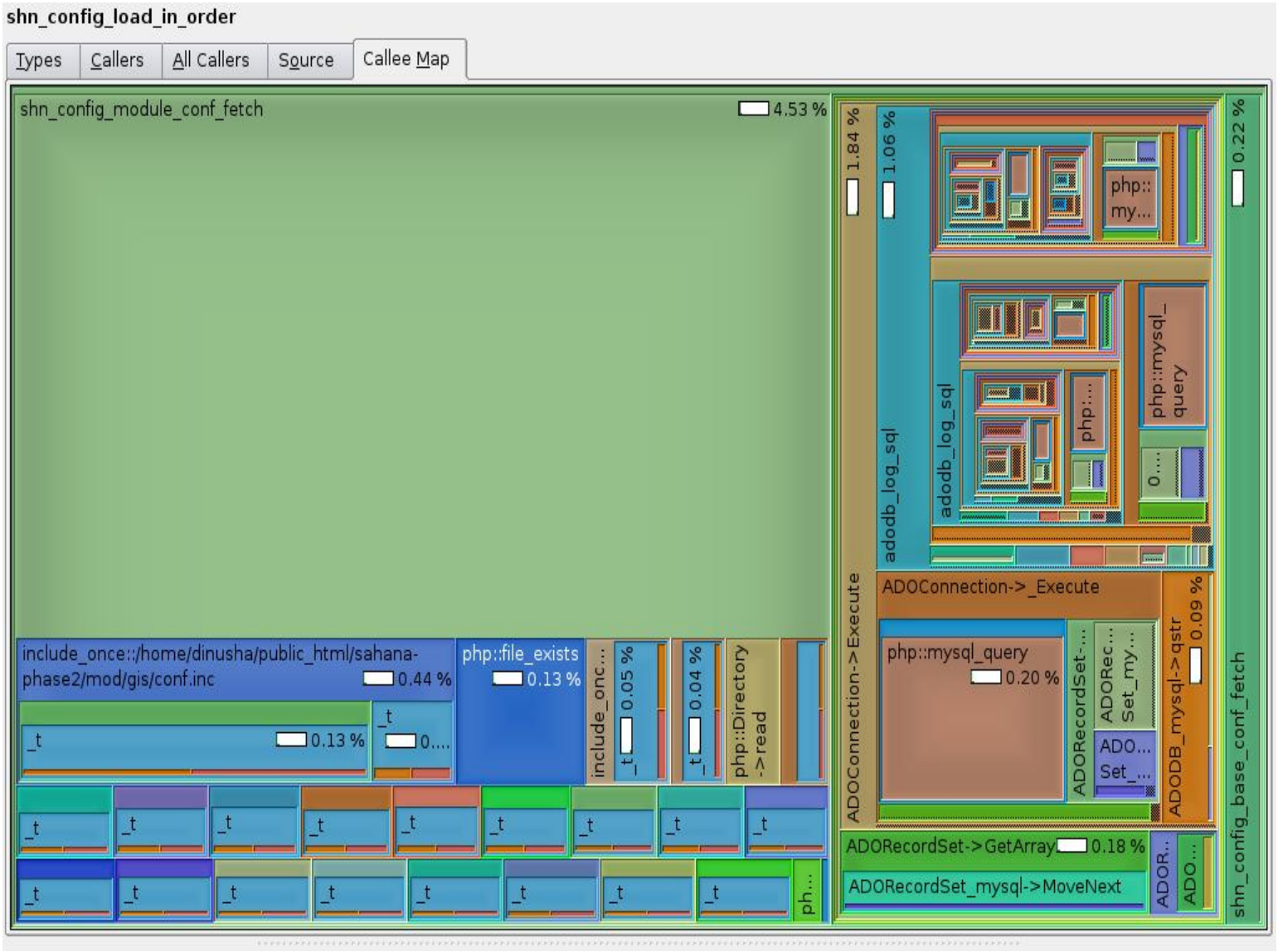


The graph view is displayed below

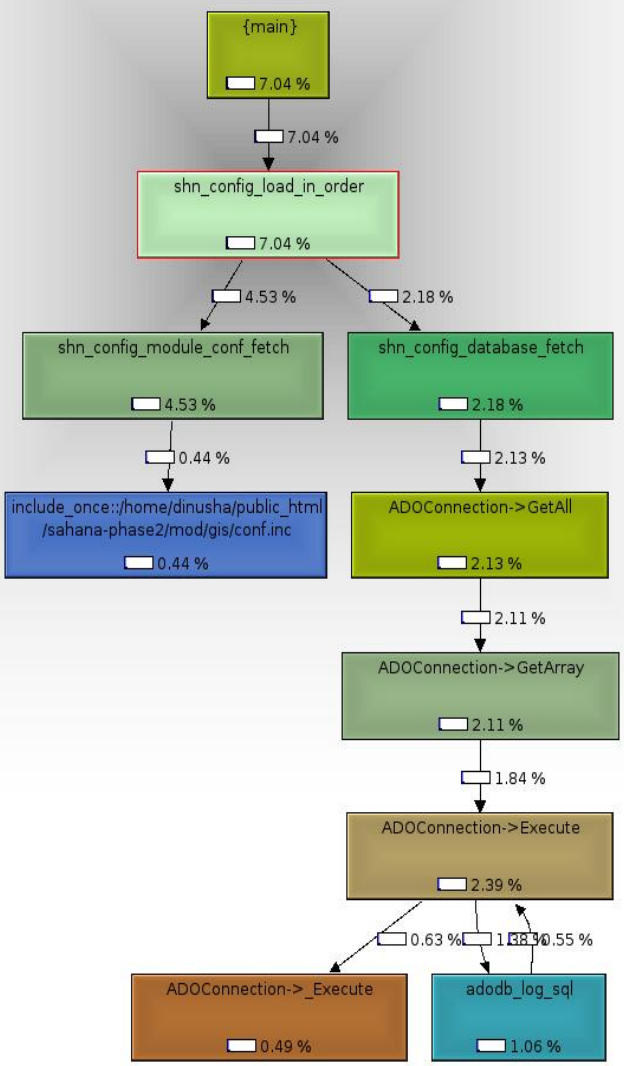


Shn_config_load_in_order

Function shn_config_load_in_order () and its children take 7.04% of the execution time. All notable functions within that is displayed below



We can see that the function shn_config_module_conf_fetch() is responsible for 4.53% of the CPU time. The graph view is displayed below.


{main}
7.04 %
7.04 %
shn_config_load_in_order
7.04 %
4.53 %
2.18 %
shn_config_module_conf_fetch
4.53 %
shn_config_database_fetch
2.18 %
0.44 %
2.13 %
include_once::/home/dinusha/public_html
/sahana-phase2/mod/gis/conf.inc
0.44 %
ADOConnection->GetAll
2.13 %
2.11 %
ADOConnection->GetArray
2.11 %
1.84 %
ADOConnection->Execute
2.39 %
0.63 %
0.55 %
ADOConnection->_Execute
0.49 %
adodb_log_sql
1.06 %

Conclusion

It is evident that HTML Purifier is taking a lot of execution time. There might be so many html violations that trigger the purifier. if we consider HTMLModuleManager.php file there are about 40 require_once which makes that thing costly. if it is possible it is better to comment some unnecessary require calls. Anyway it is not acceptable to consume about half of the computational resources for this non-functional issue. It is advisable to manually check for HTML errors before submitting page to the code base.

The time taken by the handler_db is quite acceptable since it execute a lot of queries those are mandatory to function the system properly. But most of the time is taken by ADOConnection->LogSQL() method which is not extremely important. That method logs all sql queries, time and parameters. that method can be disabled by ADOConnection->LogSQL($enable = false) command. For more information refer http://phplens.com/adodb/reference.functions.fnexecute.and.fncacheexecute.properties.html .

The function shn_config_module_conf_fetch() consume considerable cpu time because it loads the conf.inc from each module. There is a while loop which makes that function costly.