Uknow InfoHub

Test Report

BlXLRSMB Team

Contents

1 Introduction

1.1 Software Introduction

Component	Language	Environment	Lines	Remarks
fetcher	Python2	POSIX	747	Including general and user fetchers
prefilter	Python2	POSIX	54	Not including machine learning model
api-website	Python2	POSIX	426	Not including execute scripts
frontend-page	HTML/JS/CSS	POSIX	558	CSS:212 JavaScript:153 HTML:193
manage	Python2/Shell	POSIX	117	Management scripts
common	Python2	POSIX	699	Common library for all components
tests	Python2/Shell	POSIX	336	Including unit test and benchmark
sum		POSIX	2437	

1.2 Document Introduction

In the following sections of this document, we will present test scheme and results in details.

In Sec. 2, the methodology and environment the test is conducted is described. Detailed and accurate test procedure will be present in Sec. 3. Test result will be covered in Sec. 4. At last, evaluation of the test will be discussed in Sec. 5.

2 Test Scheme

2.1 Test Level

These tests call methods which simulate users' requests and determine whether whole project can respond to each request correctly. If the project gives the correct response as designed for one request, logic that get involved in generating response will be considered as tested. Therefore, this test is at API level.

2.2 Test Environment

The unitest[1] and coverage[2] module in Python and Apache Bench [3] tool are mainly used for testing, as shown as following:

[1] http://docs.python.org/2/library/unittest.html
[2] http://nedbatchelder.com/code/coverage/
[3] http://httpd.apache.org/docs/2.2/programs/ab.html

Name	Version	Purpose
Python unittest	2.7	Simulating Request for each API
Python coverage	2.7	Simulating Request for each API
ApacheBench	>=2.3	Performance Benchmark

Table 1: Software Requirements

3 Test Procedure

3.1 Testers

zxytim, ppwwyyxx, vuryleo, jiakai

3.2 Period

18:00 Dec 13th, 2013 to 18:00 Dec 14th, 2013.

3.3 Functional Tests

1. Test Case Designing First, read the API list, select out APIs that need to be tested. Then, design use cases for each API based on its function. At last, translate test cases into proper request format and add them into the test suite.

   Several cases that designed to test the robustness of the APIs are recorded as well. These cases will perform illegal requests such as requests with wrong format, permission overstep and injection.

2. Environment Setting At first, use virtualenv to build a virtual python environment for this project. Then use install tool located in manage suite to set up runtime environment for it.

   In order to provide production environment, a mongodb server and a redis server is also required.

3. Software Configuring In this test, the default configure is applied. So no interfere is needed in this step.

4. Test Configuring In order to test some third party function, correct authentication information is required. The configure file can be found easily in test suite.

5. Test Executing Execute test entry scripts to do the tests.

6. Log Analyzing After getting the test log, parse out the useful information and ordinate them into proper format.

3.4 Performance Tests

3.4.1 Test Case Designing

Several APIs that may have heavy loads are selected to get the performance data at the worse state. Then a random set of APIs are generated in order to test the average performance. Besides, APIs that are requested most frequently are selected to get the common performance.

3.4.2 Environment Setting

In propose of simulating real world performance, client and server are located on different machines, so the network transform is considered.

To do so, deploy the project in production mode on one machine, and get the test suite on another.

3.4.3 Test Executing

Execute test entry scripts to do the tests.

3.4.4 Log Analyzing

Performance test will generate plenty of log. So a data visualization procedure is taken to display the data in different collections and patterns.

4 Test Result

4.1 Functional

All functional tests are passed. For more details please refer to App. A.

4.2 Performance

Here a sample of test result is presented. For detailed all plots, see appendix.

4.3 Converge

The converge ratio reached 90%. For more details please refer to appendix converge sheet.

The code that are not converged is mainly exception handle phase. But for each kind of exception handle, at least one case is designed to test it. For example, all most all APIs are login required, but only one test case that try to logout while not logged in is designed. For one kind of exception handle procedure is the same, but may occur in many situation. Designing duplicate test cases for each API doesn't means a lot but cause plenty of time.

For others, they are designed for further widening usage that not implement yet. Such as more configurable fetcher, user defined color theme and so on.

5 Test Evaluation

5.1 Evaluation on the project

5.1.1 Functional

All APIs work correctly as expected. And all APIs is robust even under sedulous injection or other kind of inroads. This result reveals the wise design of this project at the beginning and rich knowledge of the develop team in network security field.

5.1.2 Performance

Both average performance and common performance are quite well, but there are still work to do to speed up in the worst case. However, the enhancement is limited, since in the worst case, the API will return too much of data, and the bottleneck is therefore at network bandwidth. Therefore, an API for incremental request is needed in the future. This kind of API will remember what they have provided and won't return them again if not directly requested.

Regardless of this, the worst response time is admissible by user, so further enhancement can be arrested until it really makes sense.

5.2 Evaluation on the test

5.2.1 Functional

Functional test includes legal and illegal cases, considered boundary situation and uncommon use cases. Besides, many common injection cases are used, to test the security of the APIs. This part of test cases are completed and have covered many professional assault method.

5.2.2 Performance

For more detailed result to locate the bottleneck of the system, an enhancement for the performance test is needed. Current result only shows the response time for all APIs but doesn't have enough details, such as time spent on network transform and database query.

A Test Cases

A.1 Sample

Name	Purpose	Input	Expect	Real	status
Sample	Whether the API website works	None	answer=42	answer=42	

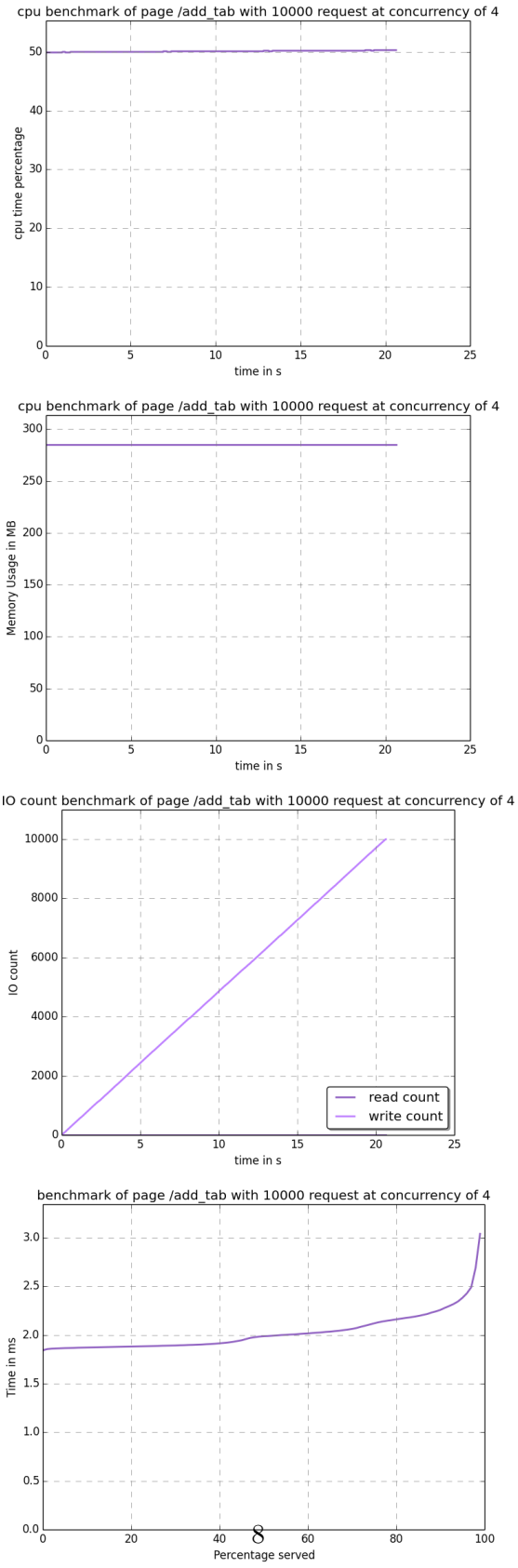


Figure 1: Test result of **add_tab** API

A.2 User Management Test

Name	Purpose	Input	Expect	Real	status
Login Fail	Login with unregistered username	Fake user	no such user	no such user	
Register	Register a new user	Unregistered user	User's information	User's information	Logged in
Register Duplicate	Register with a registered username	Register user	user already exists	user already exists	
Login	Login	Register user	Users' information	Users' information	Logged in
Logout	Logout when logged in	None	Login Page	Login Page	Logged out
Logout Fail	Logout when not logged in	None	401 error	401 error	
Login Fail	Login with incorrect password	Incorrect user	wrong password	wrong password	
Login Illegal	Login with illegal format	Illegal request	400 error	400 error	
Register Illegal	Register with illegal format	Illegal request	400 error	400 error	
Register Too Short	Register with short field	Short field	field too short	field too short	
Login Test	Determine whether logged in	Logged in required API	200	200	

A.3 Tab Test

Name	Purpose	Input	Expect	Real	status
Add Fail	Add Tab when not logged in	Tab information	401 error	401 error	
Add Fail	Add Tab with illegal format	Illegal request	400 error	400 error	
Add	Add Tab	Tab information	Tab information	Tab information	
Add	Add Tab that exists	Tab information	tab already exists	tab already exists	
Get All	Get all Tabs	None	Tab information	Tab information	
Get	Get specific Tab	Tab name	Tab information	Tab information	
Delete	Delete specific Tab	Tab name	Tab information	Tab information	
Delete	Delete not exists Tab	Tab name	Ignored	Ignored	

A.4 Tag Test

Name	Purpose	Input	Expect	Real	status
Add Fail	Add Tag when not logged in	Tag information	401 error	401 error	
Add Fail	Add Tag with illegal format	Illegal request	400 error	400 error	
Add	Add Tag	Tag information	Tag information	Tag information	
Add	Add Tag that exists	Tag information	tab already exists	tab already exists	
Get All	Get all Tags	None	Tag information	Tag information	
Get	Get specific Tag	Tag name	Tag information	Tag information	
Delete	Delete specific Tag	Tag name	Tag information	Tag information	
Delete	Delete not exists Tag	Tag name	Ignored	Ignored	

A.5 User Fetcher Test

Name	Purpose	Input	Expect	Real	status
Get All	Get all Fetcher	None	Fetcher information	Fetcher information	
Enable	Enable a Fetcher	ID and configure	Fetcher information	Fetcher information	
Refresh	Refresh Fetcher	None	Fetch result	Fetch result	
Disable	Disable a Fetcher	ID	Fetcher information	Fetcher information	

A.6 Misc

Name	Purpose	Input	Expect	Real	status
Get All Entries	Get all Entries	None	Entries information	Entries information	

A.7 Performance Test

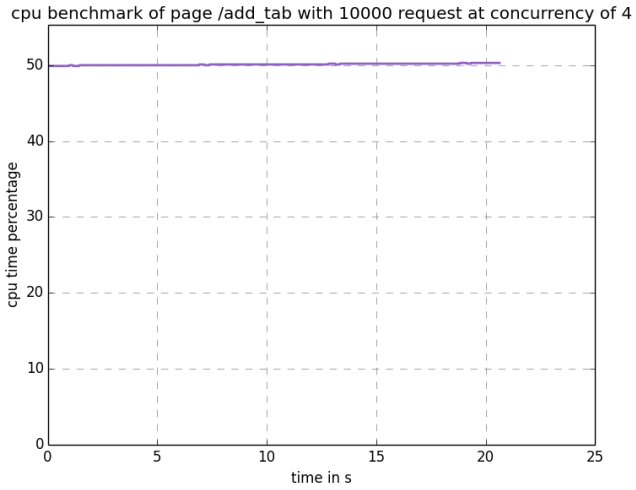
A.7.1 Heavy Load API

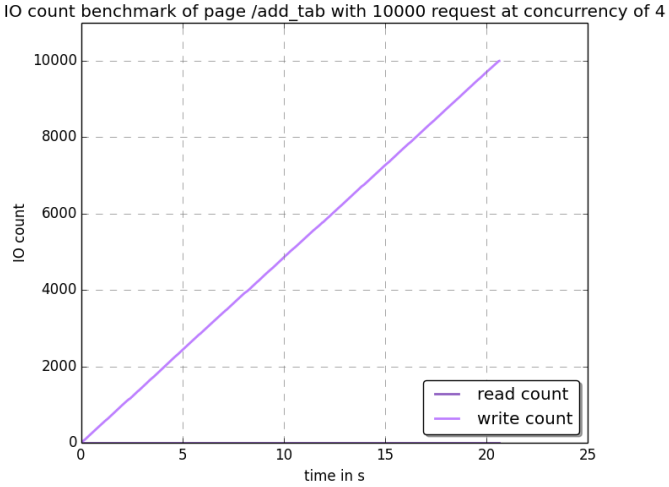
1. Get All Fetcher
2. Get All Tab
3. Get All Tag
4. Get All Entries

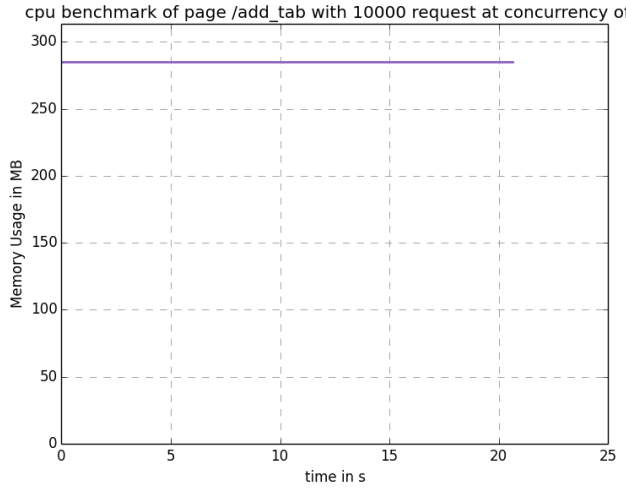
A.7.2 Common API

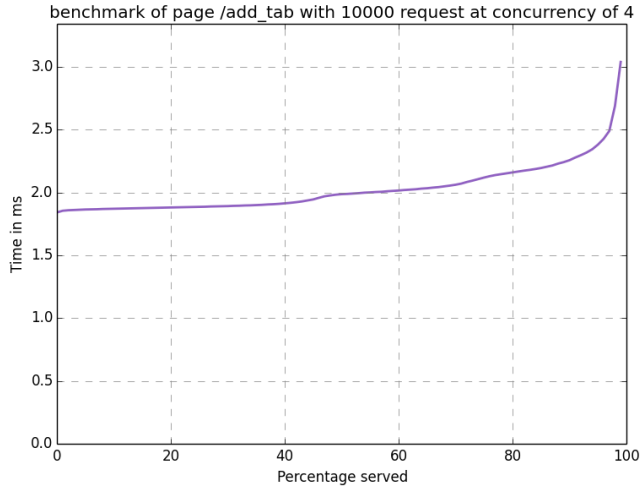
1. Login
2. Register
3. Logout
4. Get All Entries

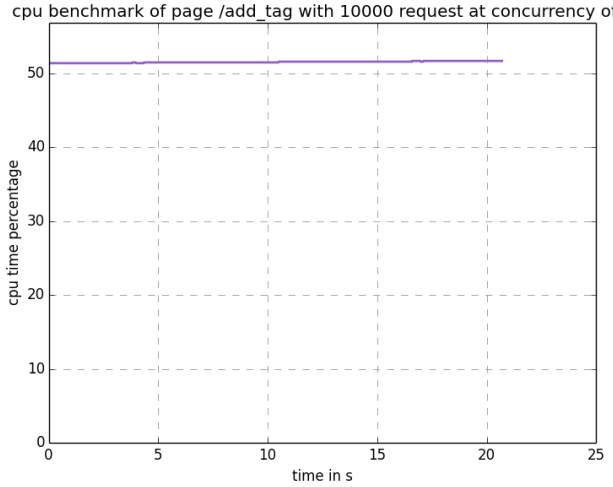
B Benchmark Result

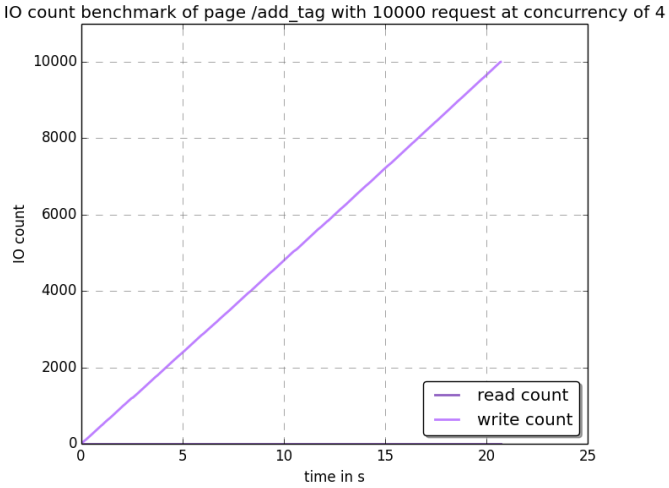


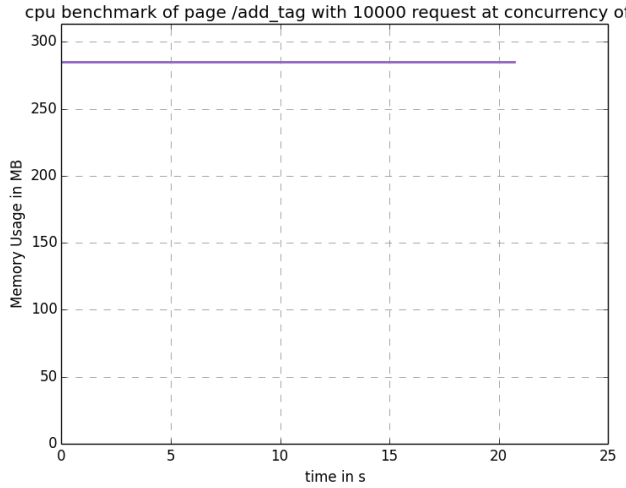

IO count benchmark of page /add_tab with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

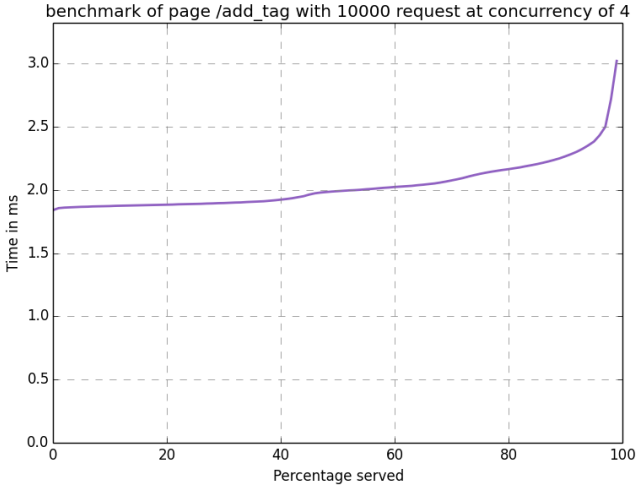

cpu benchmark of page /add_tab with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

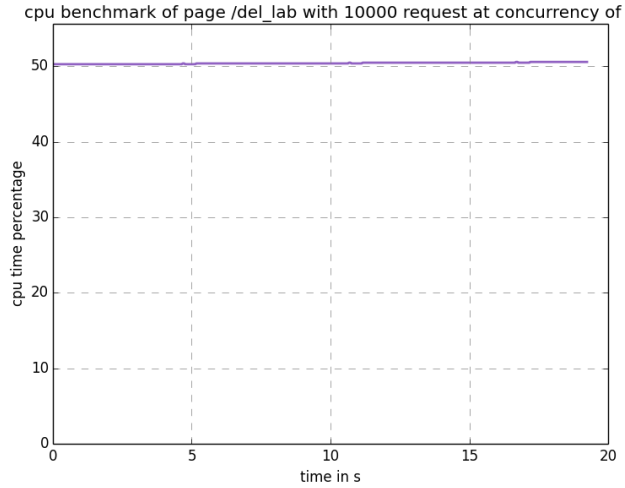

benchmark of page /add_tab with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

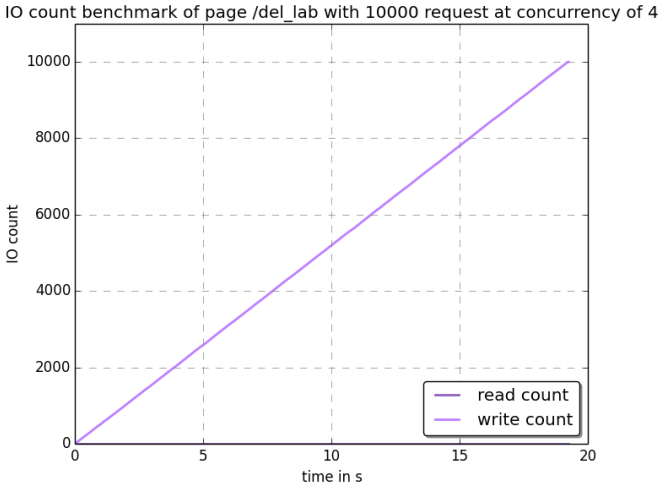

cpu benchmark of page /add_tag with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

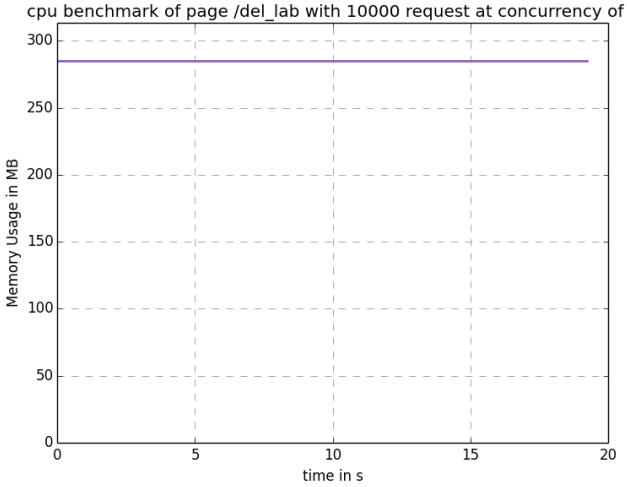

IO count benchmark of page /add_tag with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

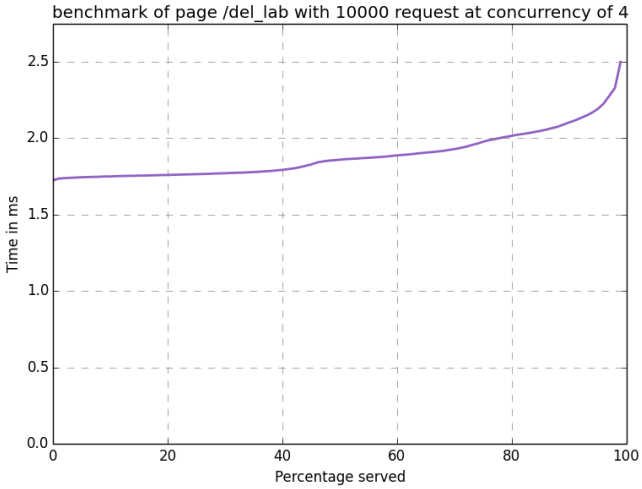

cpu benchmark of page /add_tag with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

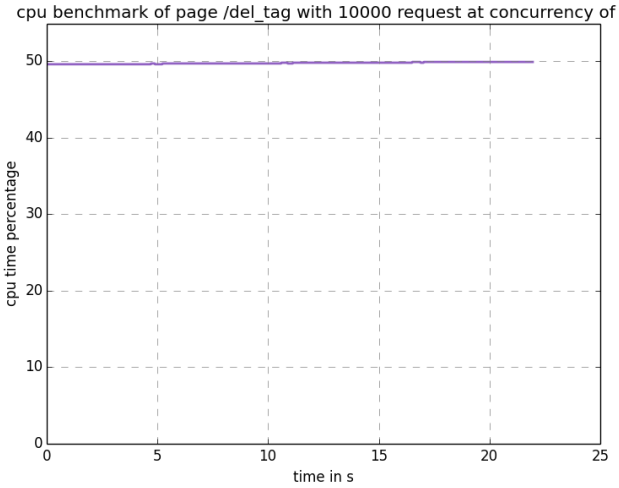

benchmark of page /add_tag with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

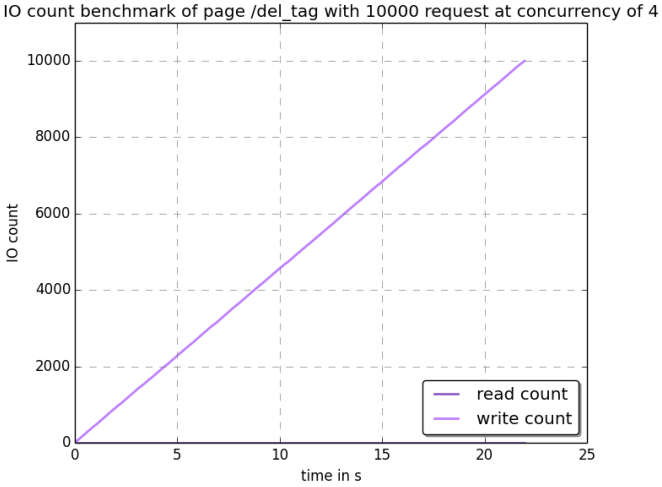

cpu benchmark of page /del_lab with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20

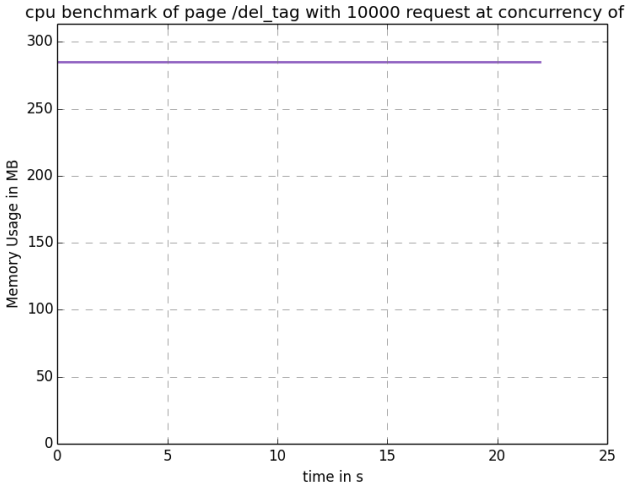

IO count benchmark of page /del_lab with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20

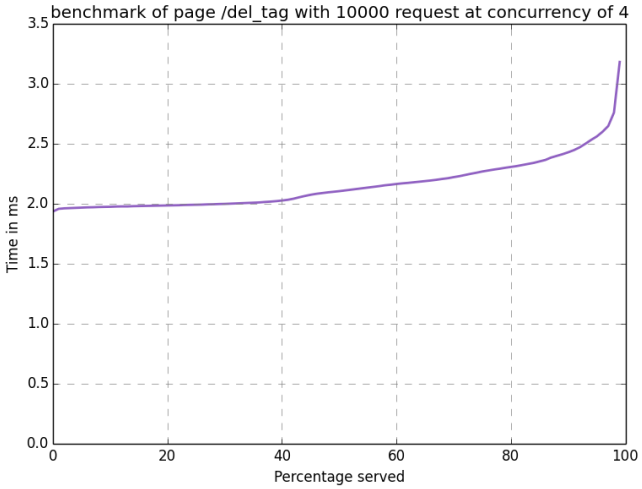

cpu benchmark of page /del_lab with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20

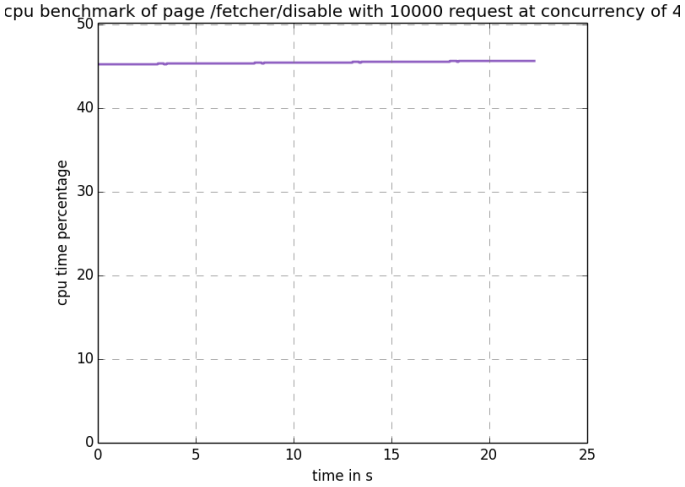

benchmark of page /del_lab with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
0
20
40
60
80
100

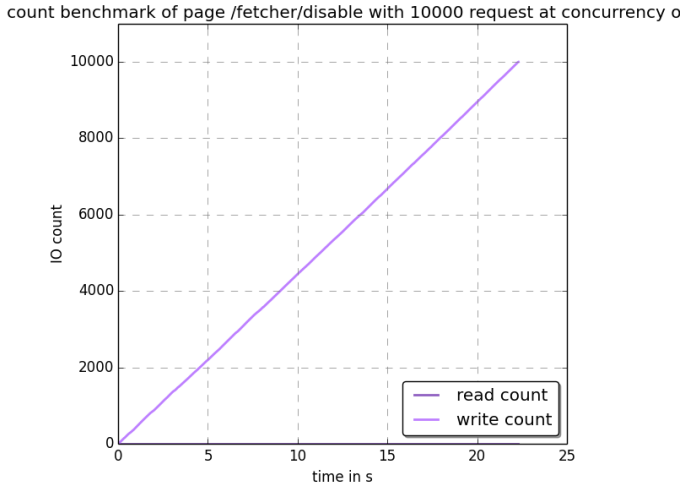

cpu benchmark of page /del_tag with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

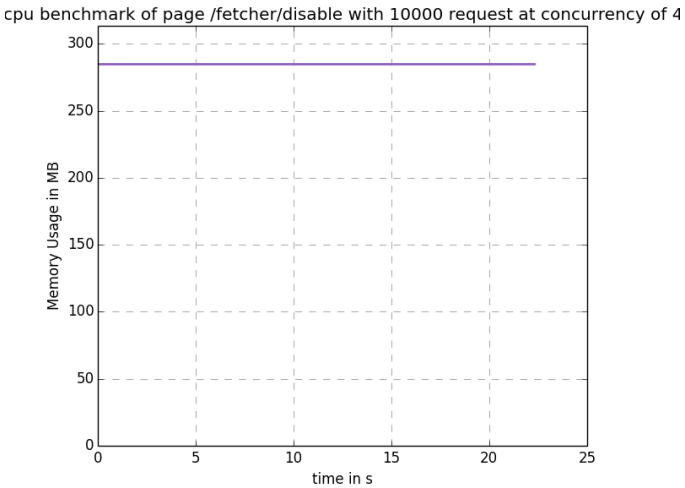

IO count benchmark of page /del_tag with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

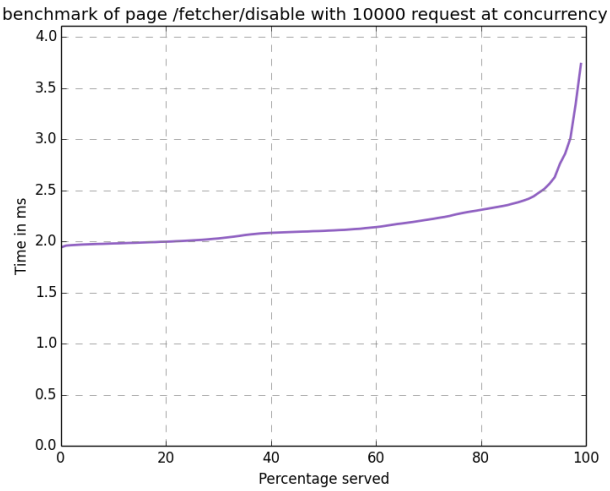

cpu benchmark of page /del_tag with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

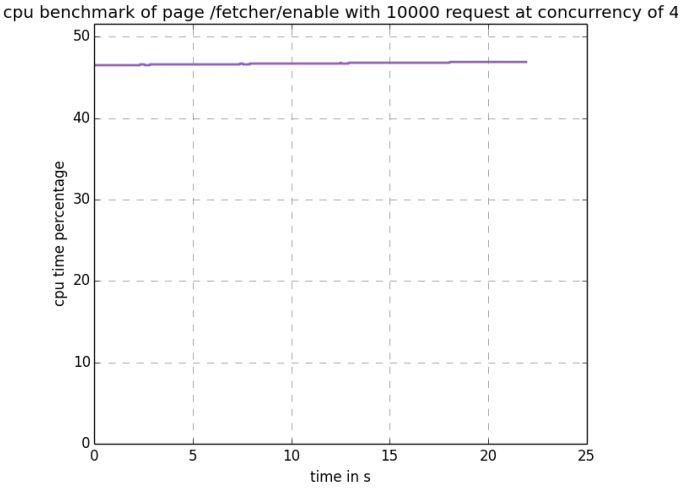

benchmark of page /del_tag with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
20
40
60
80
100

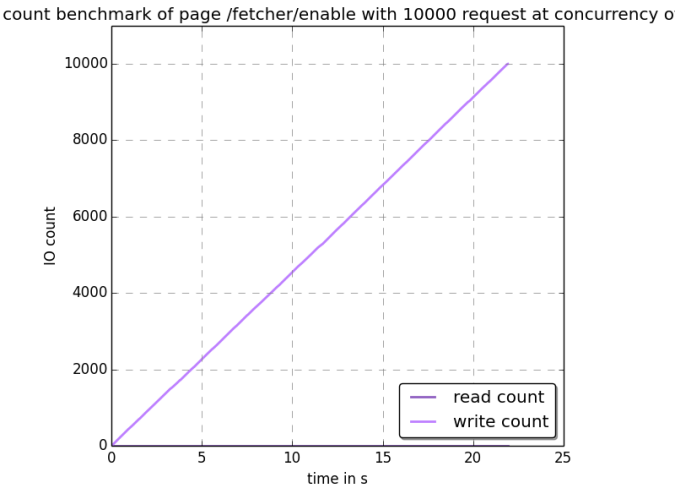

cpu benchmark of page /fetcher/disable with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

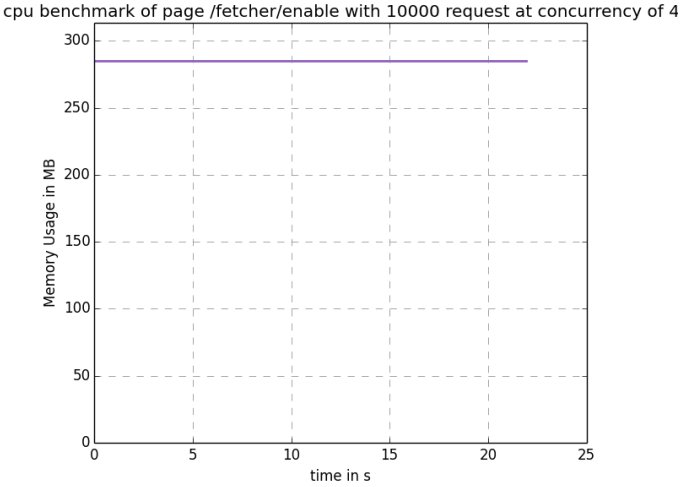

count benchmark of page /fetcher/disable with 10000 request at concurrency o
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

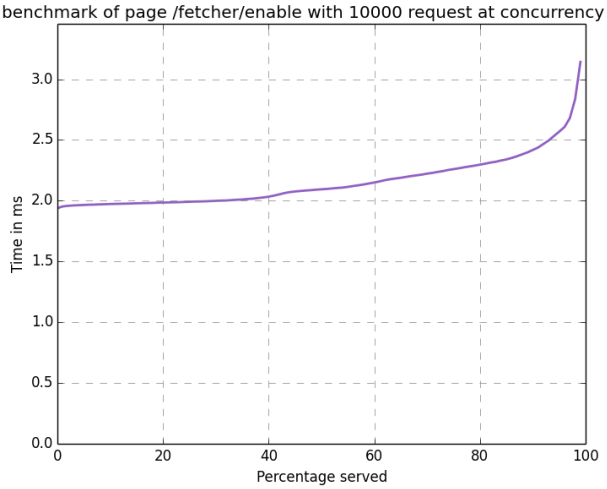

cpu benchmark of page /fetcher/disable with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

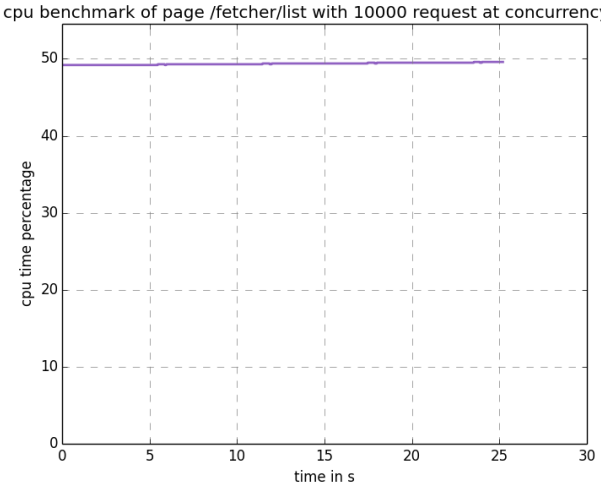

benchmark of page /fetcher/disable with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
20
40
60
80
100

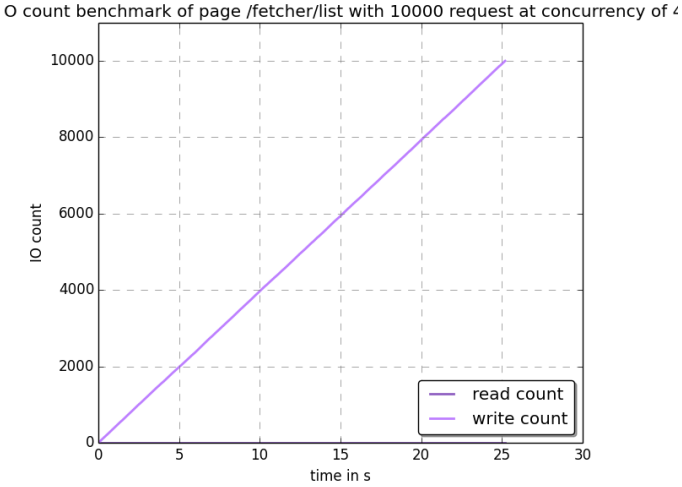

cpu benchmark of page /fetcher/enable with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

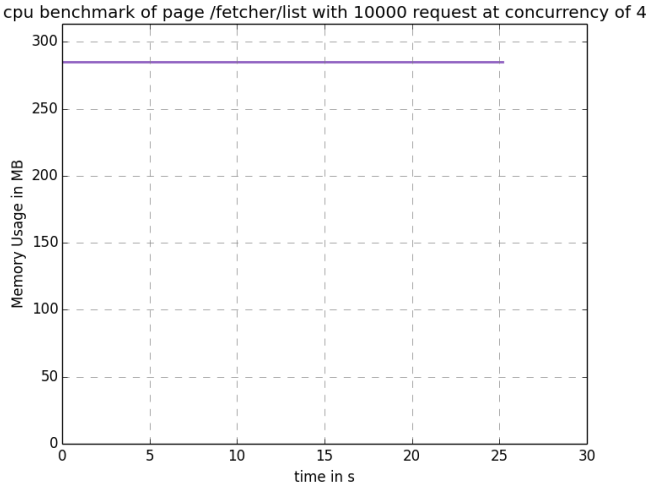

count benchmark of page /fetcher/enable with 10000 request at concurrency o
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

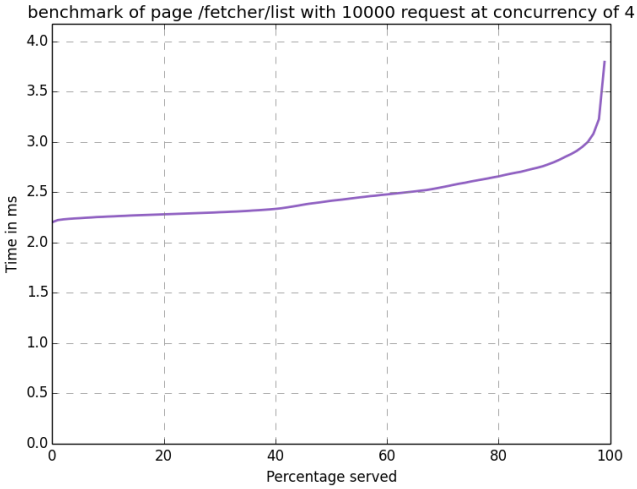

cpu benchmark of page /fetcher/enable with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

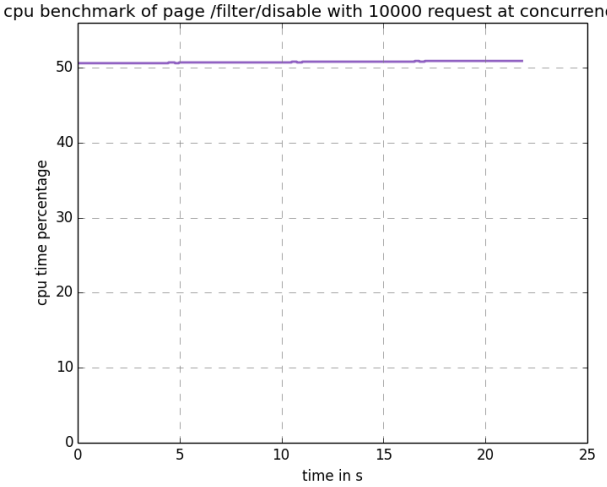

benchmark of page /fetcher/enable with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

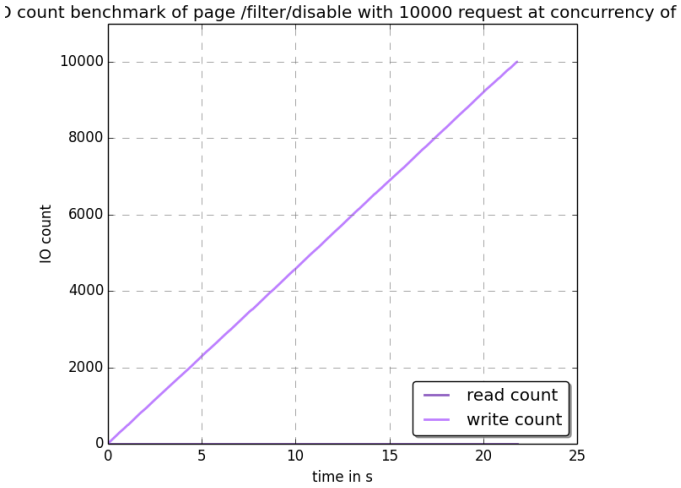

cpu benchmark of page /fetcher/list with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25
30

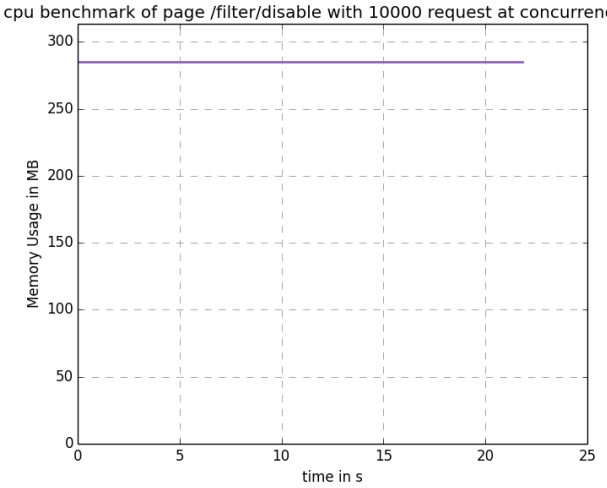

O count benchmark of page /fetcher/list with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25
30

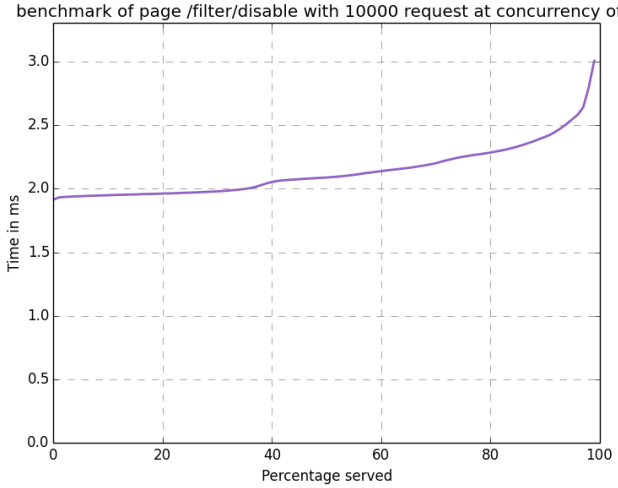

cpu benchmark of page /fetcher/list with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25
30

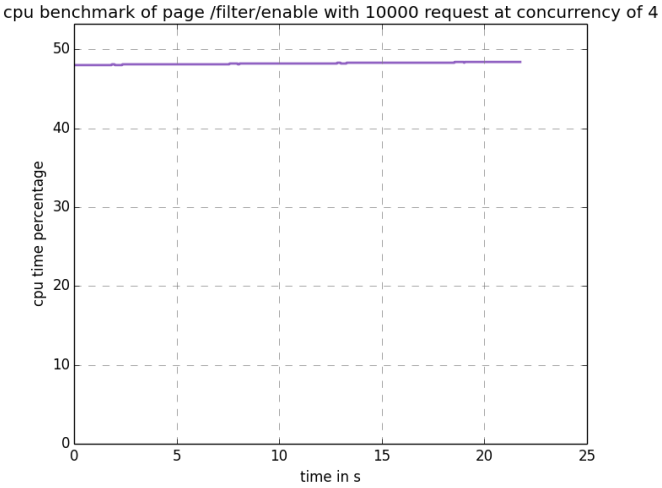

benchmark of page /fetcher/list with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
20
40
60
80
100

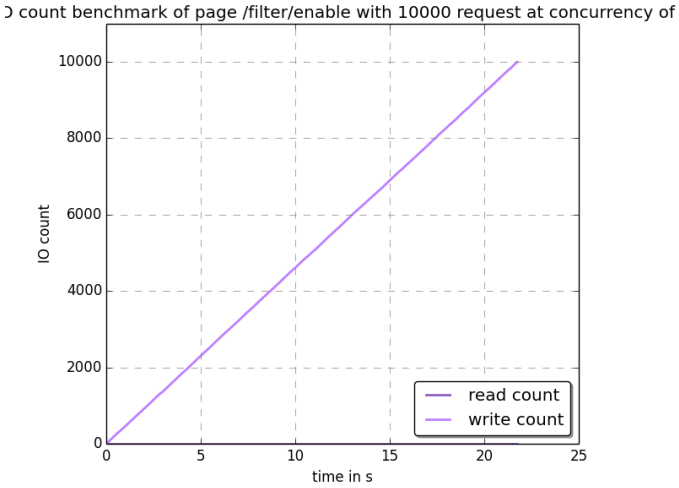

cpu benchmark of page /filter/disable with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

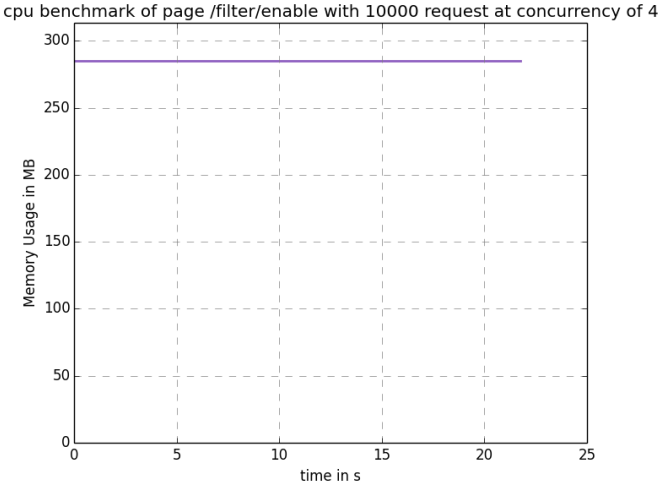

count benchmark of page /filter/disable with 10000 request at concurrency of
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

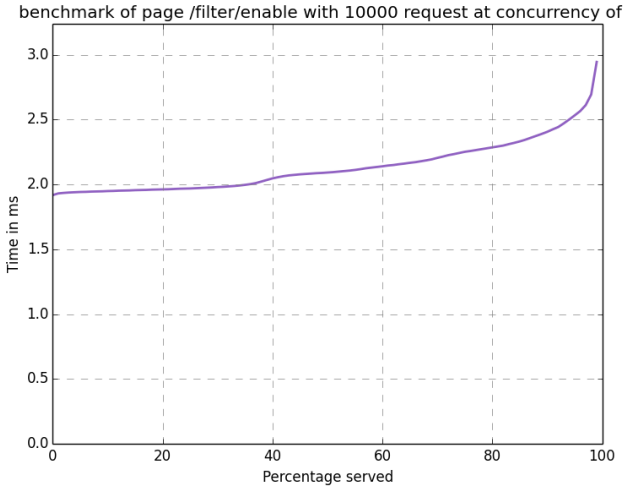

cpu benchmark of page /filter/disable with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

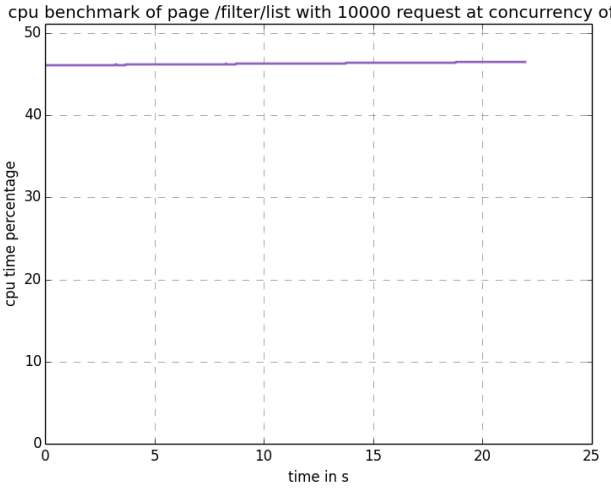

benchmark of page /filter/disable with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

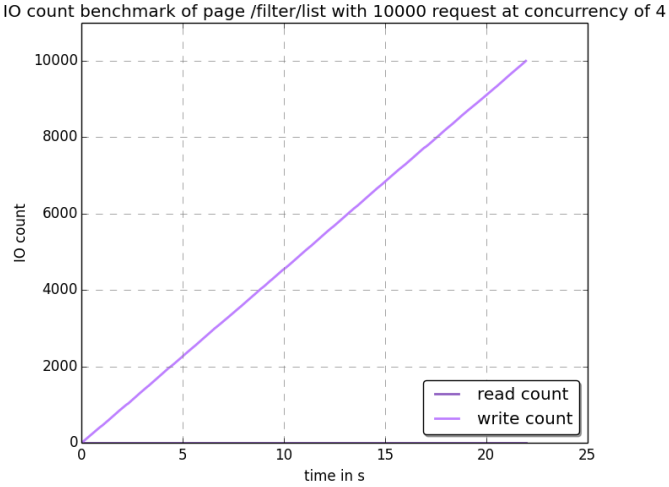

cpu benchmark of page /filter/enable with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

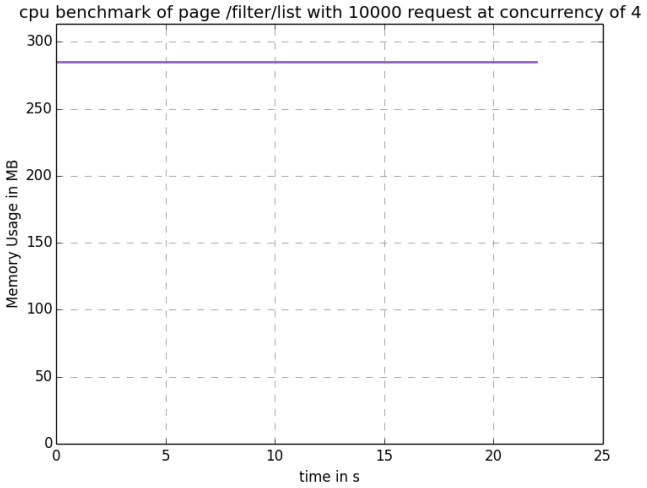

O count benchmark of page /filter/enable with 10000 request at concurrency of
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

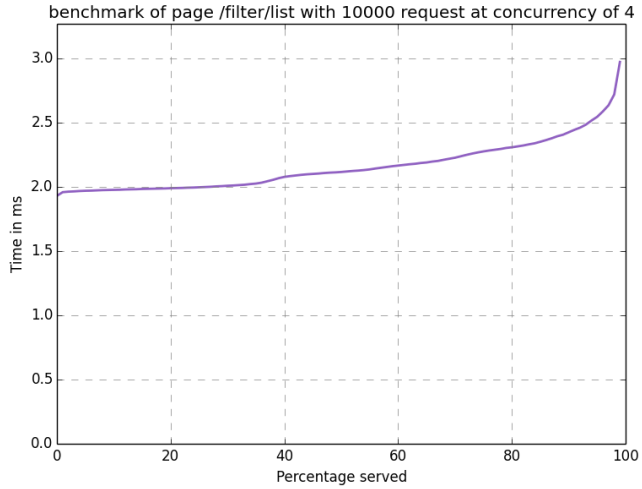

cpu benchmark of page /filter/enable with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

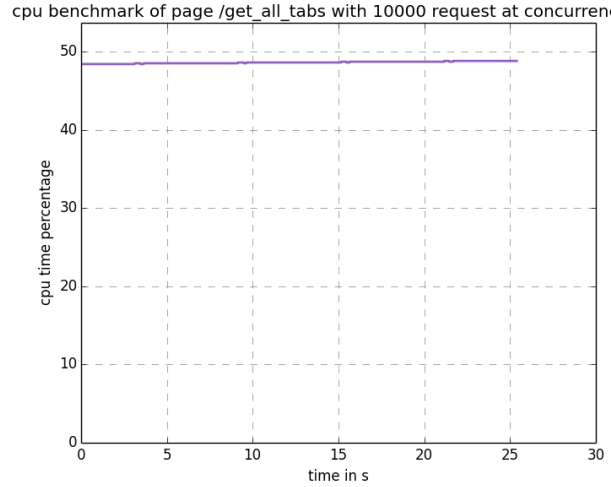

benchmark of page /filter/enable with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

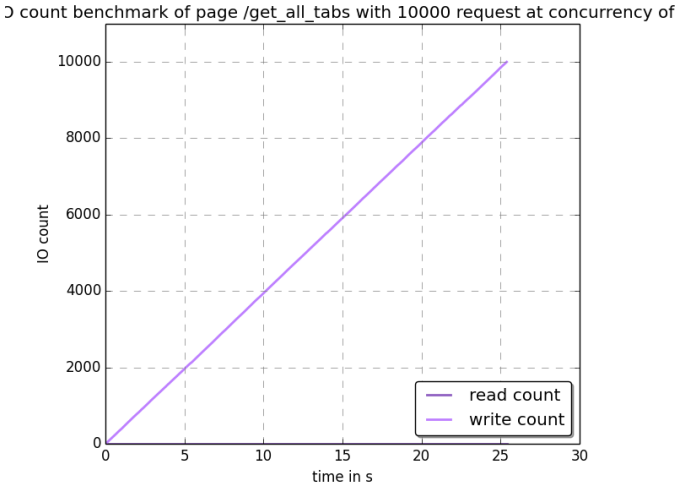

cpu benchmark of page /filter/list with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

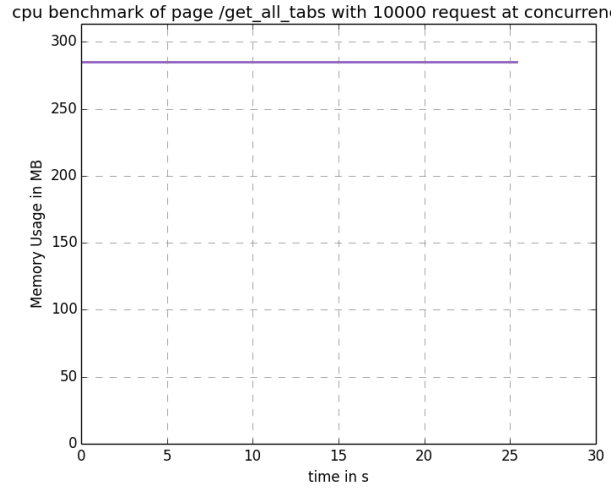

IO count benchmark of page /filter/list with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

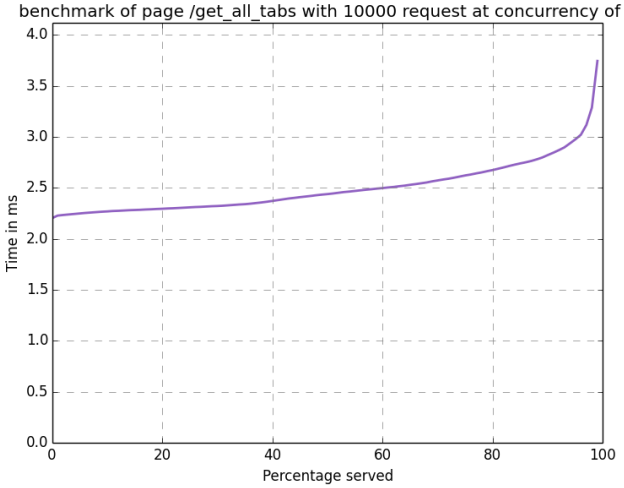

cpu benchmark of page /filter/list with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

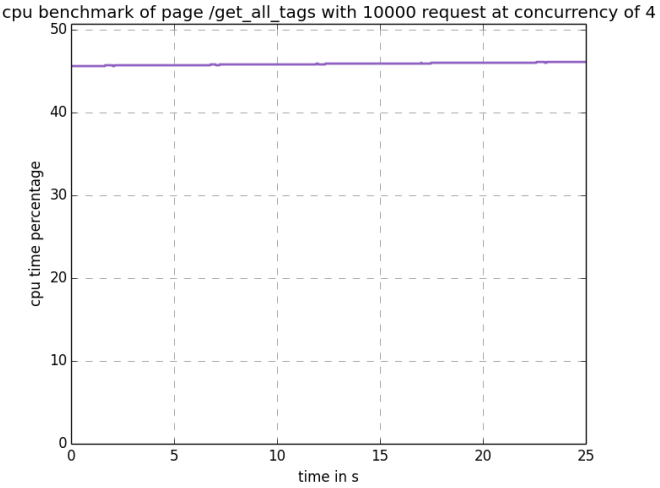

benchmark of page /filter/list with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

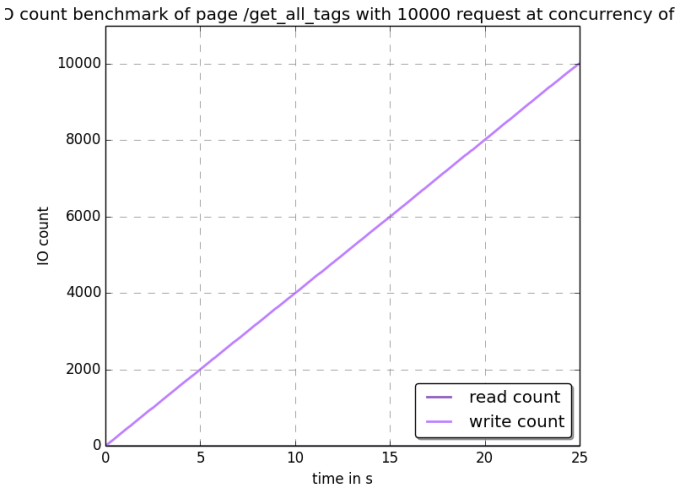

cpu benchmark of page /get_all_tabs with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25
30

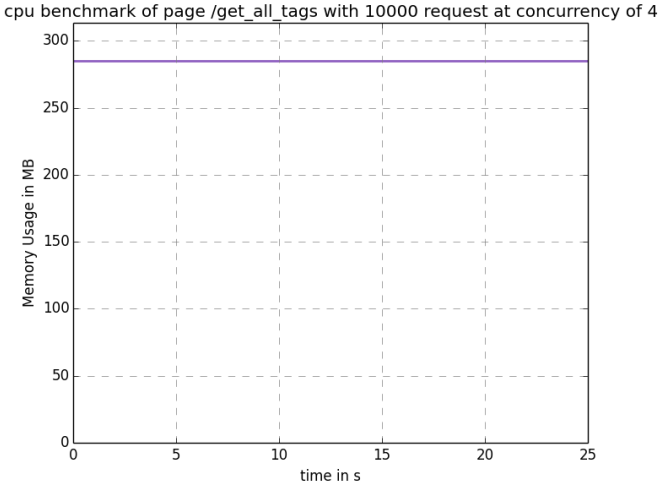

Ɔ count benchmark of page /get_all_tabs with 10000 request at concurrency of
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25
30

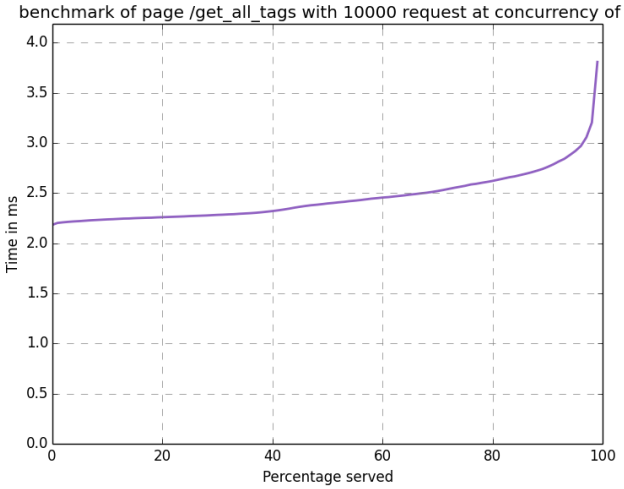

cpu benchmark of page /get_all_tabs with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25
30

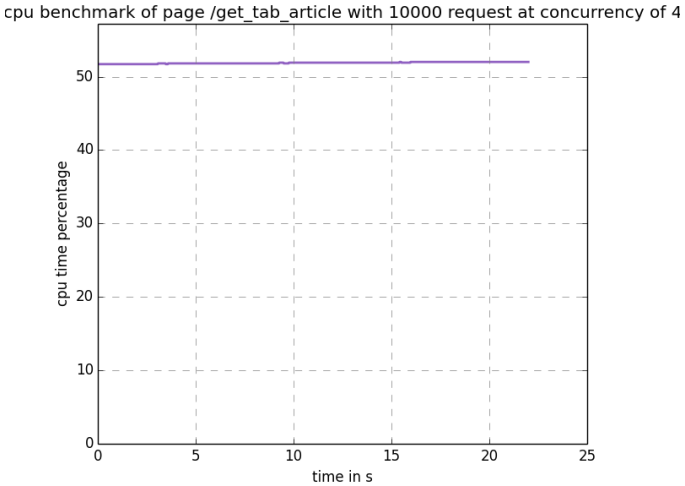

benchmark of page /get_all_tabs with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
20
40
60
80
100

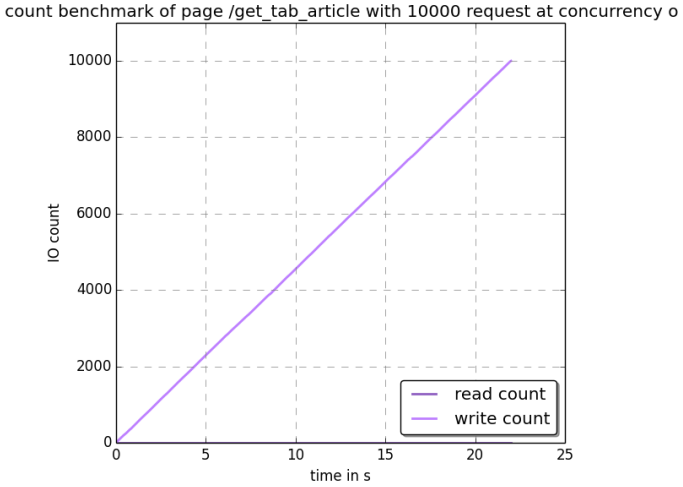

cpu benchmark of page /get_all_tags with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

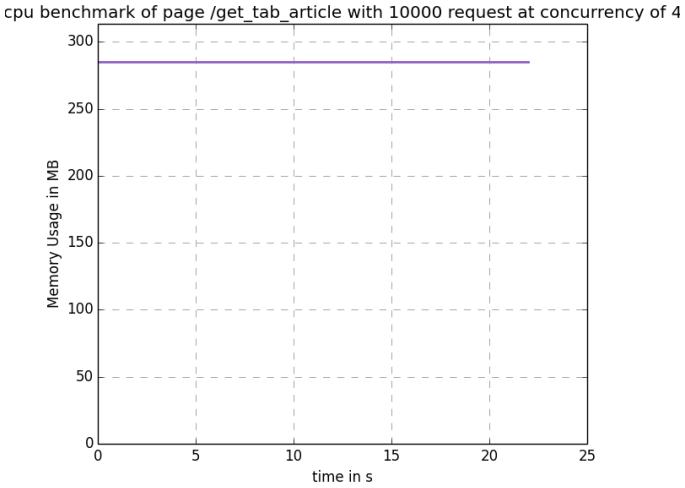

O count benchmark of page /get_all_tags with 10000 request at concurrency of
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

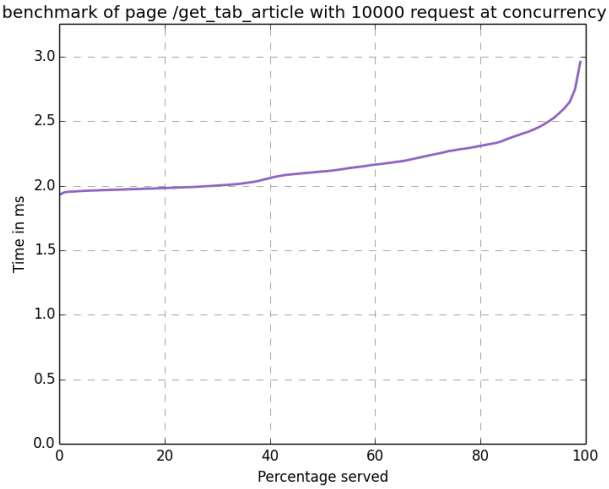

cpu benchmark of page /get_all_tags with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

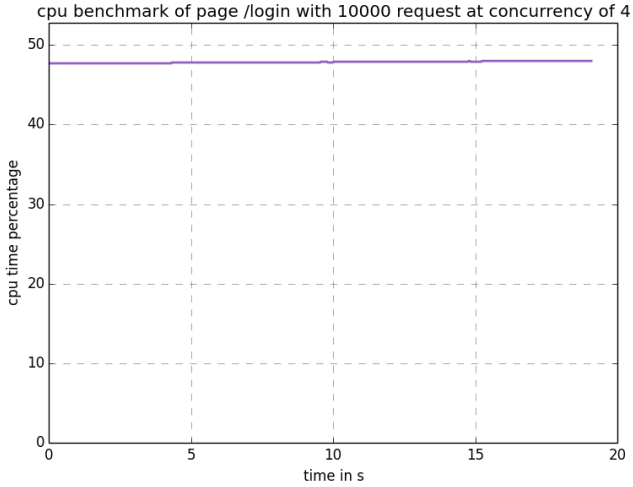

benchmark of page /get_all_tags with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
20
40
60
80
100

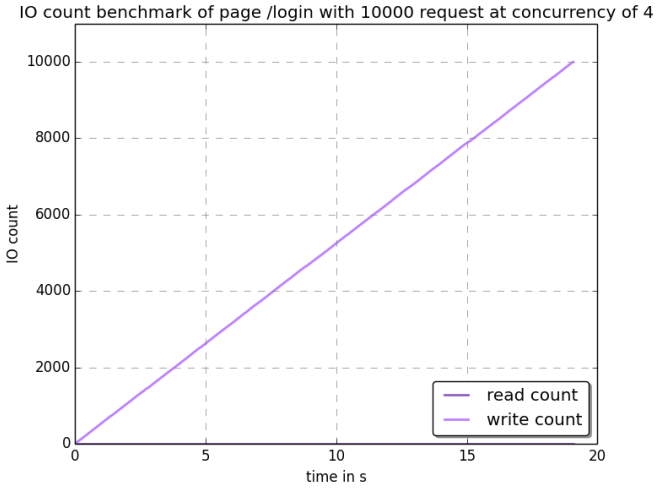

cpu benchmark of page /get_tab_article with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

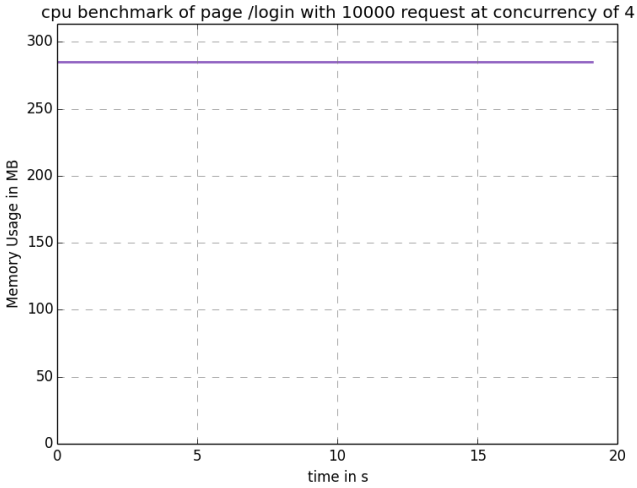

count benchmark of page /get_tab_article with 10000 request at concurrency o
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

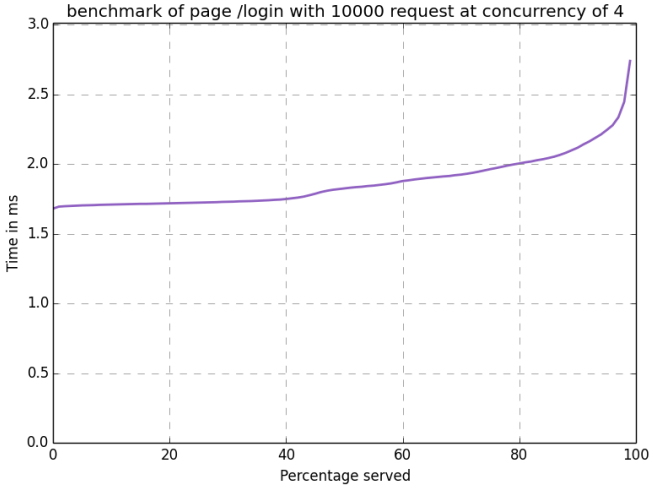

cpu benchmark of page /get_tab_article with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

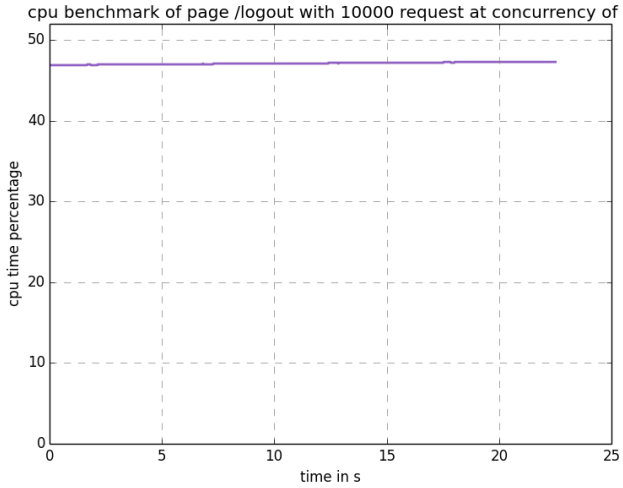

benchmark of page /get_tab_article with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

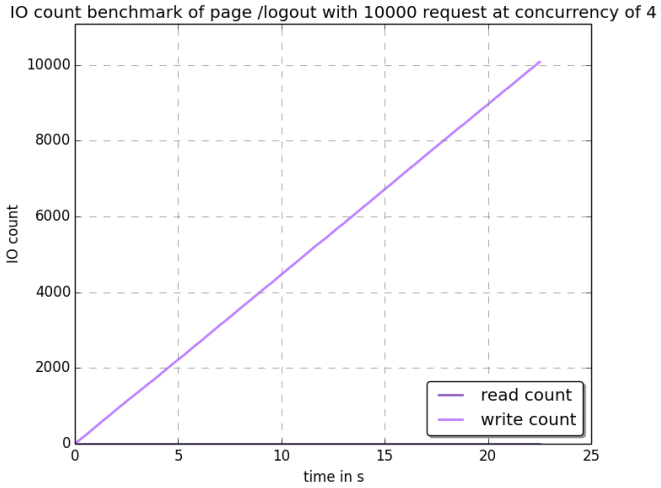

cpu benchmark of page /login with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20

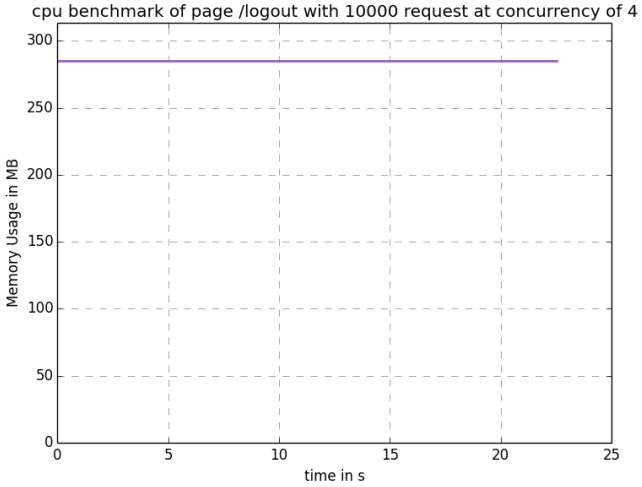

IO count benchmark of page /login with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20

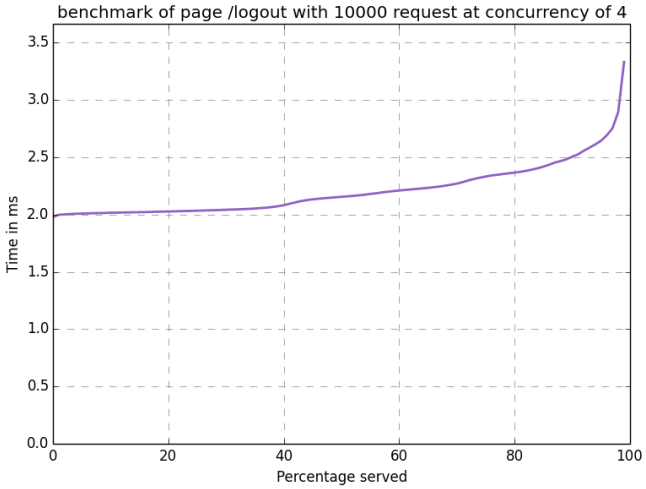

cpu benchmark of page /login with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20

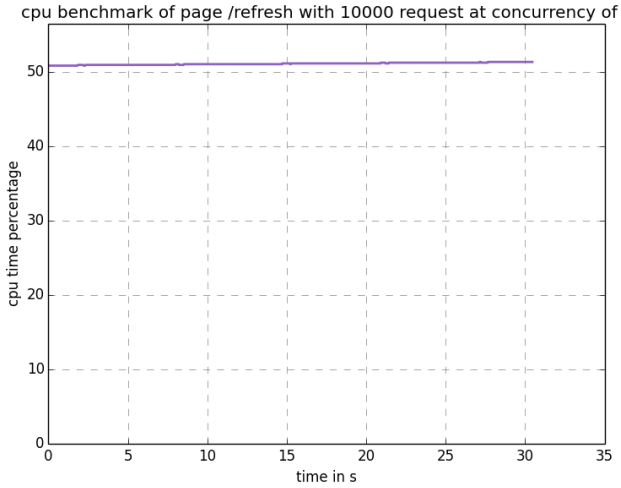

benchmark of page /login with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

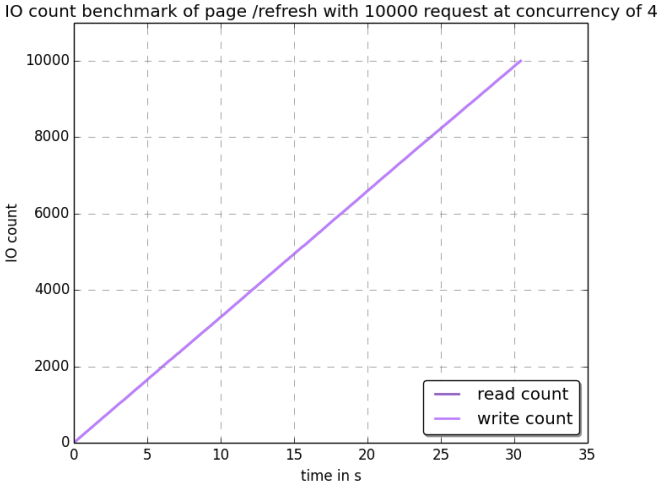

cpu benchmark of page /logout with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25

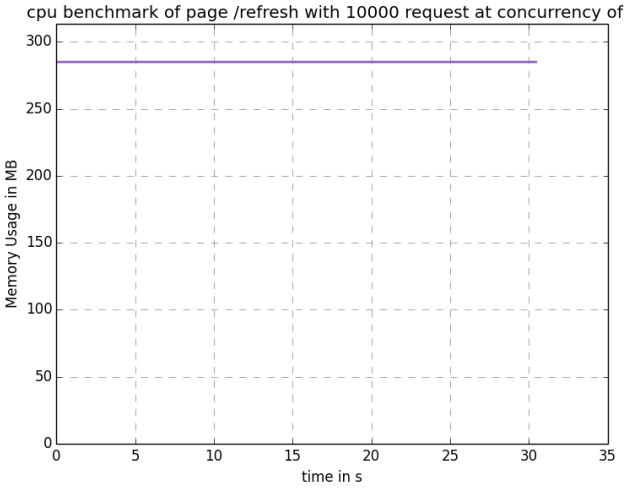

IO count benchmark of page /logout with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25

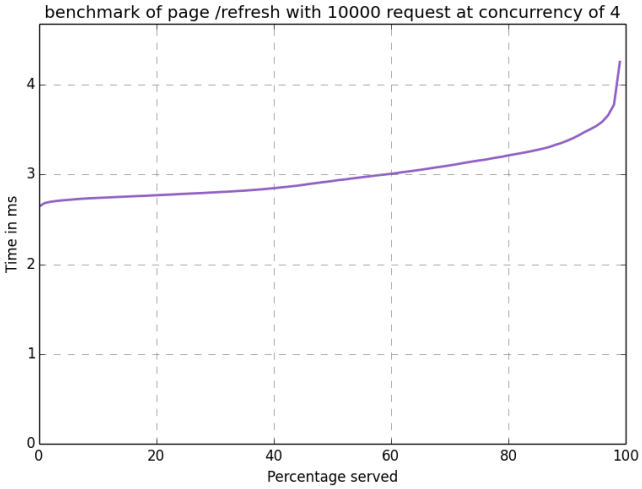

cpu benchmark of page /logout with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25

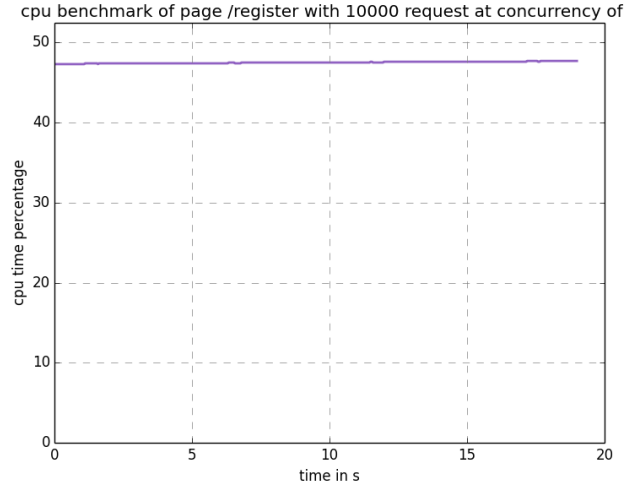

benchmark of page /logout with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
20
40
60
80
100

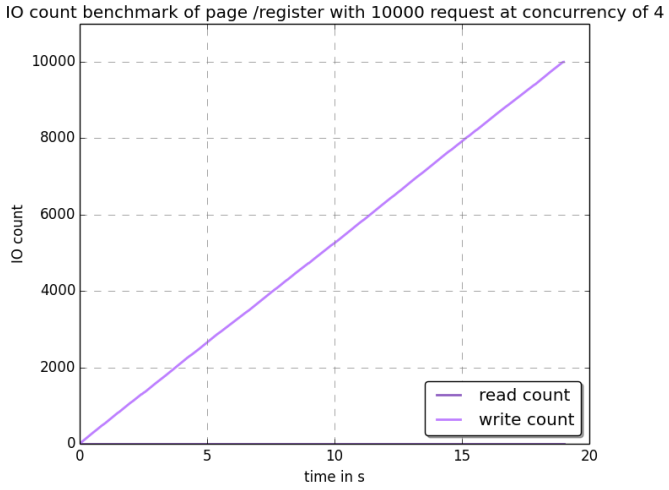

cpu benchmark of page /refresh with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25
30
35

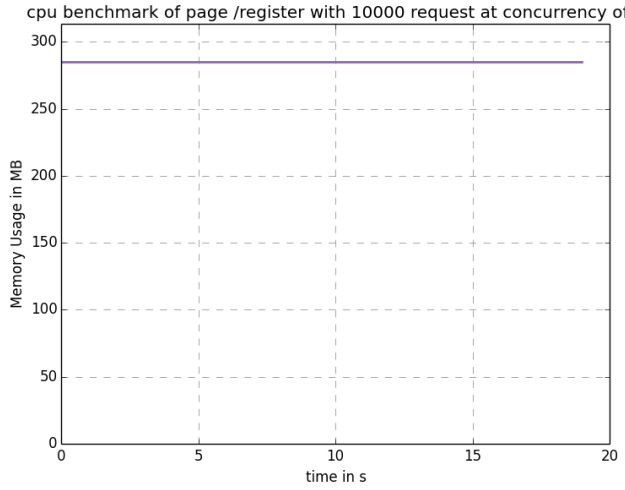

IO count benchmark of page /refresh with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25
30
35

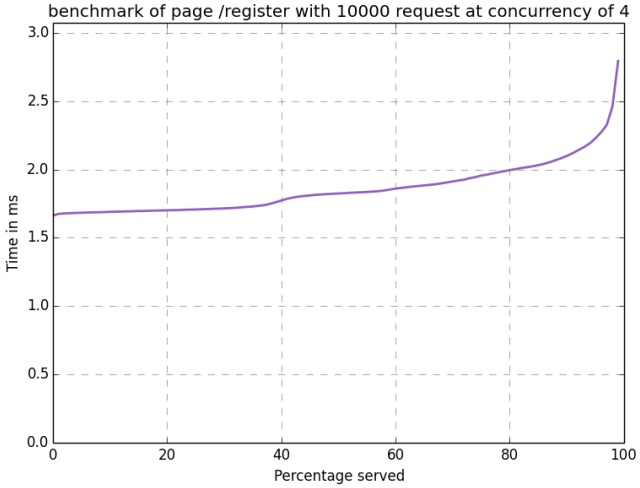

cpu benchmark of page /refresh with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25
30
35

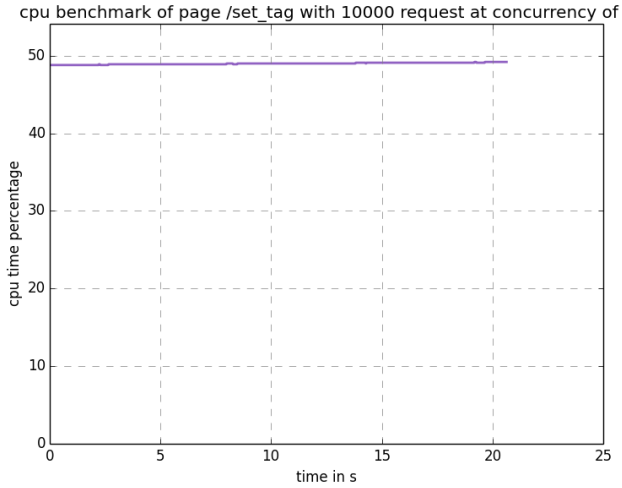

benchmark of page /refresh with 10000 request at concurrency of 4
Time in ms
Percentage served
0
1
2
3
4
0
20
40
60
80
100

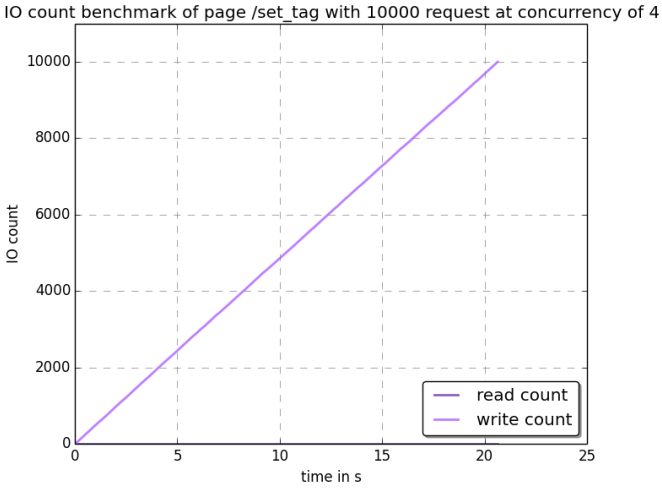

cpu benchmark of page /register with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20

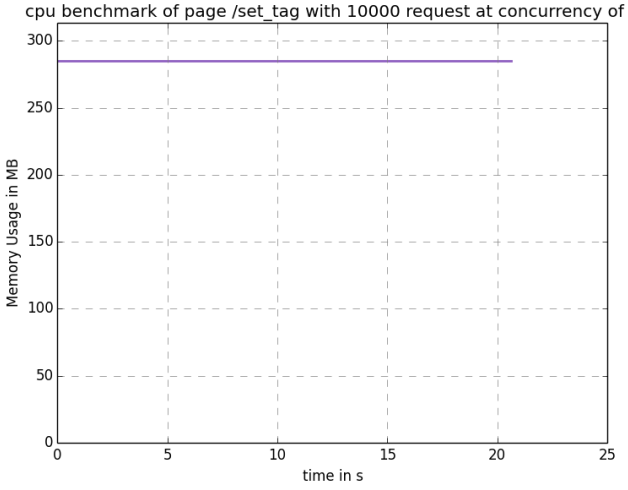

IO count benchmark of page /register with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20

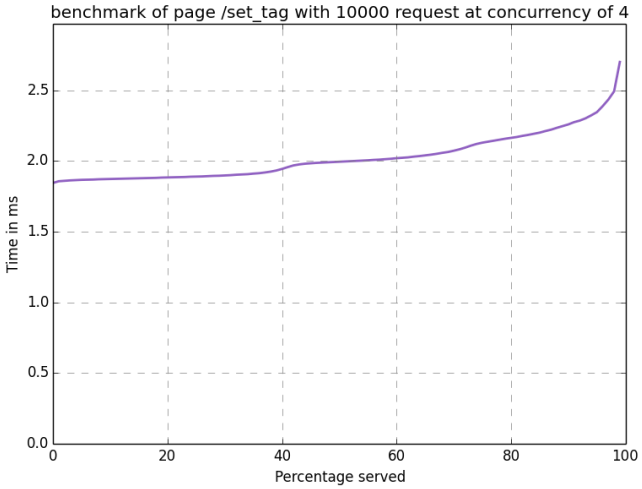

cpu benchmark of page /register with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20


benchmark of page /register with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100


cpu benchmark of page /set_tag with 10000 request at concurrency of 4
cpu time percentage
time in s
0
10
20
30
40
50
0
5
10
15
20
25


IO count benchmark of page /set_tag with 10000 request at concurrency of 4
IO count
time in s
read count
write count
0
2000
4000
6000
8000
10000
0
5
10
15
20
25


cpu benchmark of page /set_tag with 10000 request at concurrency of 4
Memory Usage in MB
time in s
0
50
100
150
200
250
300
0
5
10
15
20
25


benchmark of page /set_tag with 10000 request at concurrency of 4
Time in ms
Percentage served
0.0
0.5
1.0
1.5
2.0
2.5
0
20
40
60
80
100