Implementing next generation web server in Galaxy

Wai Yi Leung [1], Leon Mei [1,2]

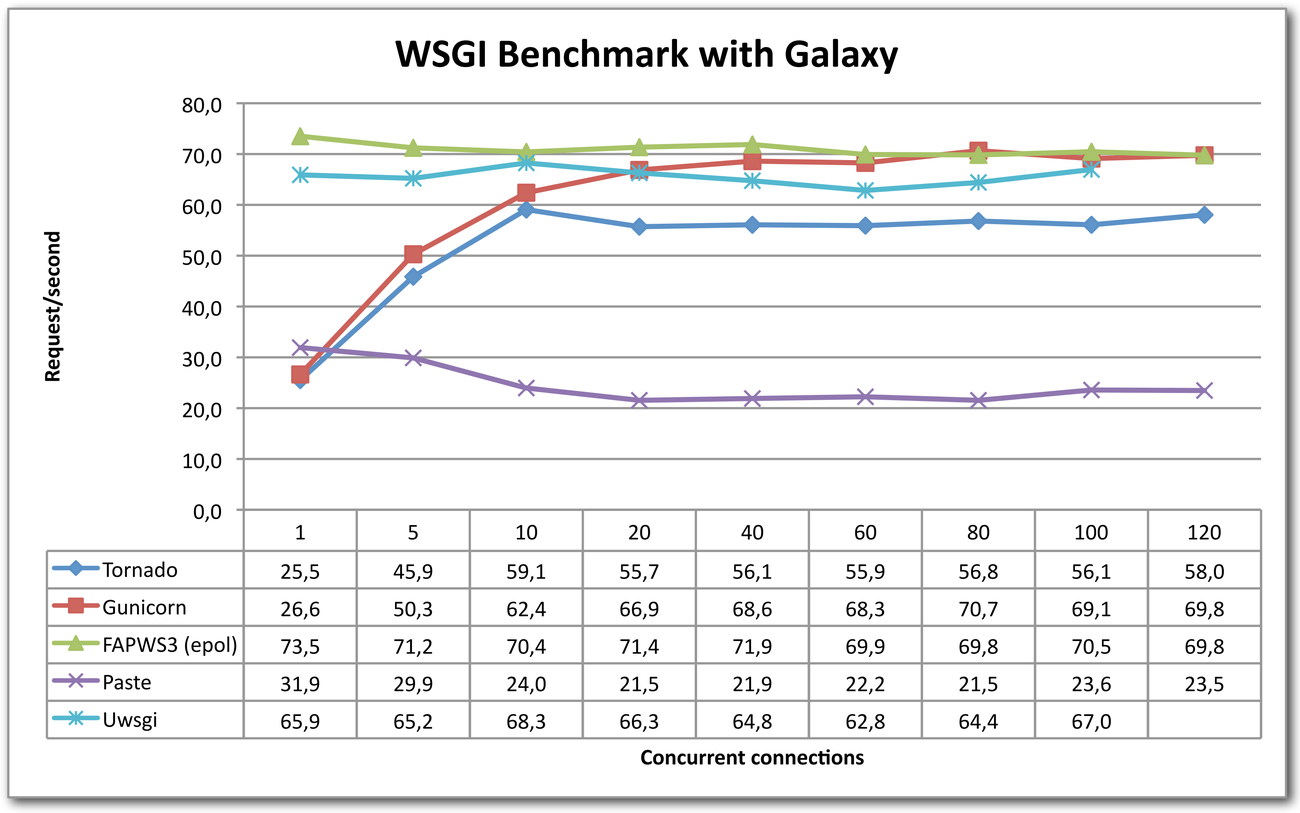
[1] Leiden University Medical Centre, Sequence Analysis Support Core, Leiden, The Netherlands
[2] Netherlands Bioinformatics Centre, Nijmegen, The Netherlands
w.y.leung@lumc.nl / github.com/wyleung

Introduction

Our interest is to see whether we can push Galaxy to a new limit on serving more request per second. The reason for this is simple: web request are (relatively) not cpu intensive. Web request are mostly database-connection and/or file-i/o bound because of the web-templates.

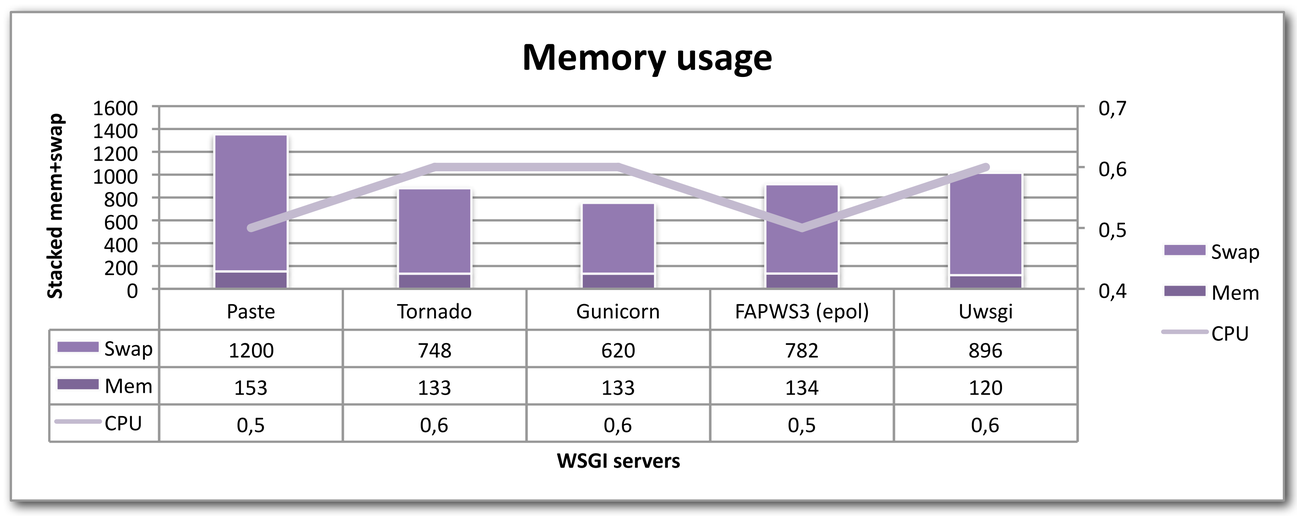
We expect performance gains with modern WSGI-servers compared to the default setup with the Paste-server for Pylons.

A secondary goal of this project is to provide a package for sysadmins to easily change the WSGI server in their setup.



	1	5	10	20	40	60	80	100	120
Tornado	25,5	45,9	59,1	55,7	56,1	55,9	56,8	56,1	58,0
Gunicorn	26,6	50,3	62,4	66,9	68,6	68,3	70,7	69,1	69,8
FAPWS3 (epol)	73,5	71,2	70,4	71,4	71,9	69,9	69,8	70,5	69,8
Paste	31,9	29,9	24,0	21,5	21,9	22,2	21,5	23,6	23,5
Uwsgi	65,9	65,2	68,3	66,3	64,8	62,8	64,4	67,0	

Picture 1: Benchmarking the connectivity of WSGI-servers in a Galaxy setup. Numbers for the Paste are not reliable for $n > 1$.



	Paste	Tornado	Gunicorn	FAPWS3 (epol)	Uwsgi
Swap	1200	748	620	782	896
Mem	153	133	133	134	120
CPU	0,5	0,6	0,6	0,5	0,6

Picture 2: Measurement of system resource usage. Numbers taken from top/htop while tests were performed.

```
[server:main]

use = egg:Paste#http

# The replacements:

#use = egg:galaxyservers#tornado
#use = egg:galaxyservers#gunicorn
#use = egg:galaxyservers#fapws3
```

Code listning 1: Change the WSGI server by just changing 1 line of code in your configuration. Installable using "pip install galaxyservers"

Material and methods

Two virtual machines with 2 virtual cores on an Intel i5-3570K, 8GB ram and 50GB SSD was setup for this benchmark. Both VMs were connected to a GBit-network.

Both machines were installed with Ubuntu 12.04 LTS server edition with the base tools. The webserver had additional packages installed as required by a clean installation of Galaxy (bitbucket revision 8870:09c81e81952d). Server settings were tuned to accept high number of connections and open sockets.

A similar setup was applied on the client-machine where we conducted the benchmark. We choose to use AB v2.3 (Apache Bench) for the benchmarking.

Each WSGI server was installed with the default installation instructions from the package. The configuration was tuned for production usage, e.g. logging and debugging options were turned off.

Discussion

In the initial setup we started testing with 1 concurrent connection. We didn't see improvements by using Tornado or Gunicorn. Only Fapws3 could significantly outperform Paste by 230%.

To simulate real world applications we open multiple connections to the webserver (e.g. pictures, ajax-sniplets etc.). We tried to profile concurrent connections by opening more connections to the server per client.

We observed not only gain in the number of connections, but also a reduction of memory and swap usage compared to the Paste profile. In multi-concurrency mode, we also noticed errors in serving pages using Paste. Paste was not able to open multiple sessions at the same time. About 280/10.000 connections were dropped because of a failure.

As these results were meassured in a closed lab-environment, we cannot really tell whether real-world applications would benefit from changing the WSGI server. Ideally this test should be tested in a course where more participants can attend

We created a python-package called **galaxyservers,** which is installable from the PyPi repository and available on GitHub for those managing high traffic Galaxy servers.

Command to install galaxyservers:
$> ***pip install galaxyservers***

or checkout the source at:
https://github.com/wyleung/galaxyservers