Designing a scalable and distributed application

EuroPython 2015
Geo distributed page hit counter web application
#1
Multi Datacenter
spans over multiple and distant datacenters
#2
A unique page hit count
sum of the page hits from all the datacenters
Automatic resizing

no manual application (re)configuration
Distributed configuration

immediately available to all web services
Hell...

...that’s some kind of contract!
be lazy!

Rely on your stack
Zen of python

if the technology is hard to explain, it's a bad idea
UNIX tools philosophy
keep it small and simple
Isolated components
breaking it down to pieces
Microservices

distributed architecture style
micromanagement = high operational cost
Communication reliability

network latency and failure
Locality vs consistency

data access lag or outage
feature rich application stack

nginx + uWSGI
- Collector gets HTTP requests produces jobs
- Collector returns the total hit count
consumes jobs -> increments a counter

processor
beanstalkd

simple and reliable job queue server
[uwsgi]
master = true
socket = /run/uwsgi/%n.socket

# daemon
attach-daemon2 = pidfile=/run/beanstalkd.pid,cmd=beanstalkd -b /var/lib/beanstalkd -p 11300 -l 0.0.0.0 -z 65536 -u beanstalk, daemonize=1
Service discovery
automated scaling and fault tolerance
Consul

multi datacenter + key / value storage
uWSGI consul plugin

seamless service registration / health check
Let’s build this up to victory
Live demo!

(hopefully)
Thanks

source code: github.com/ultrabug/ep2015
@ultrabug