



Building High Performance Sites Nginx, MySQL, Varnish and Php

Tamas Kozak
Percona Live London 2011

Introduction

- IT Director at Percona
- Scaling one of the most visited (top10) site in Hungary
- Worked at an alexa top50 company before joining Percona

Agenda

- How to serve 2 million pageviews / server / day
- Surviving “slashdot” effects
- Scaling existing and new sites with Varnish, Nginx, MySQL and PHP-FPM with minimal modifications
- Backend optimization only

Why?

- Faster page loads increase user experience
- Optimizing / rewriting code is often expensive or not an option
- Easily handle daily peak time
- Lowering datacenter costs

Key components

- Varnish cache
 - Reverse proxy, flexible configuration with inline C support
- Nginx
 - Event based
 - Lightweight
 - Runs more than 8% of the web (Netcraft)
- PHP-FPM
 - Runs in standalone mode
 - Best FastCGI implementation available for PHP
- MySQL or Percona Server

Lets put it together

Varnish Cache
Listens on public ip:80

Nginx
Listens on localhost:80

PHP-FPM

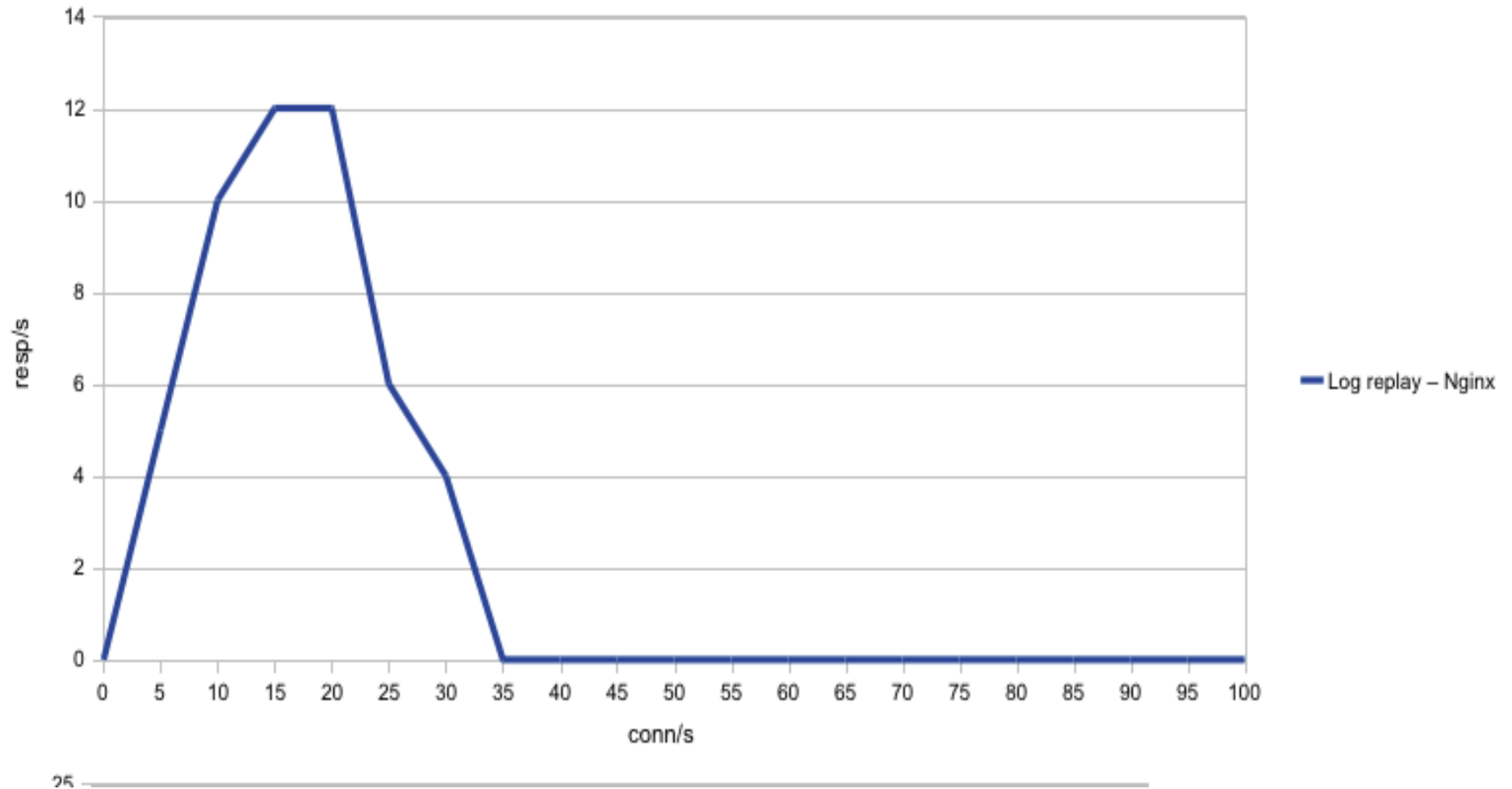
Database



About our “testing” app

- Legacy CMS written by an external contractor
- Heavy pages that are very expensive to generate
- Source code available but written many years ago
- Millions of visitors, thousands of editors

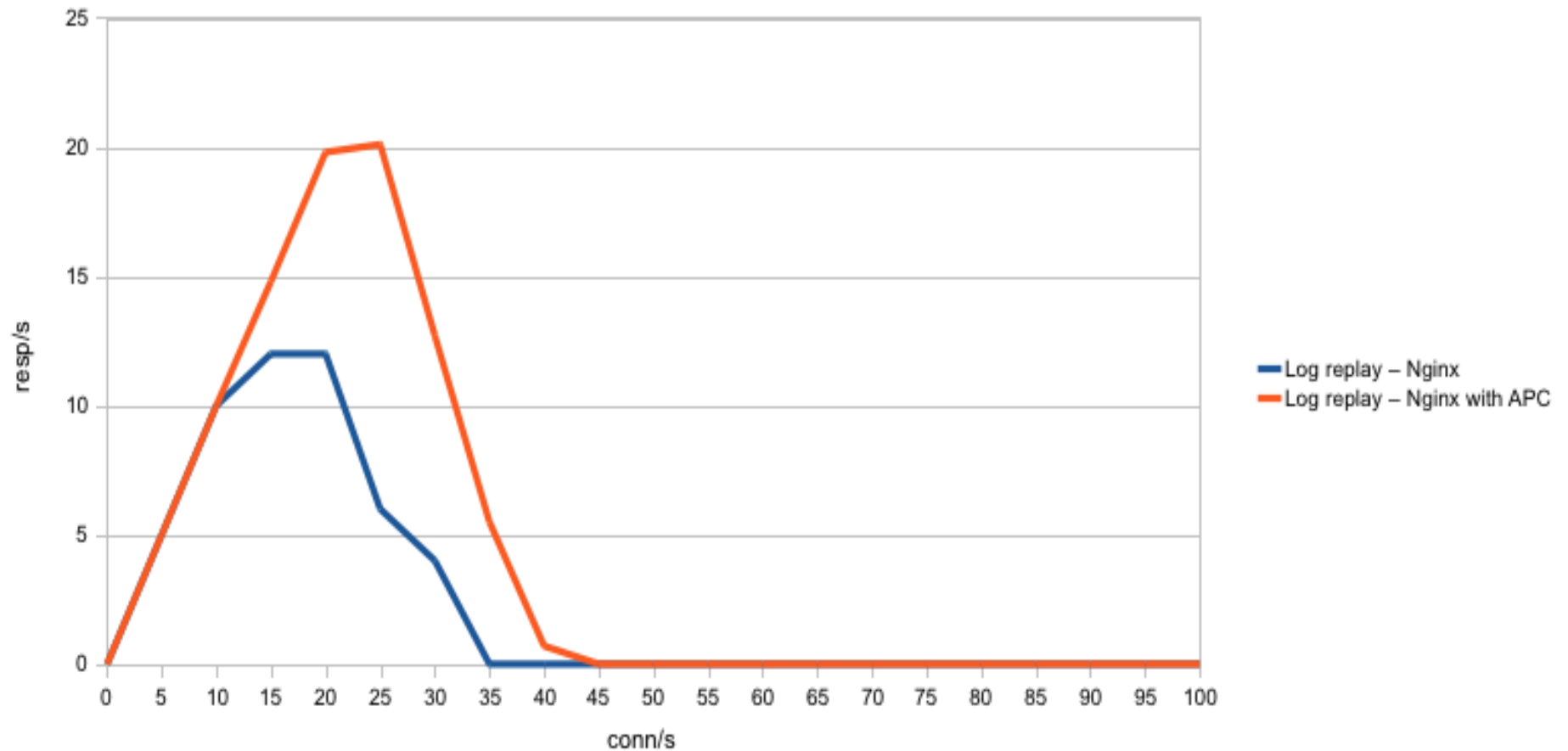
Benchmark without caching first



Fine tune defaults

- Find the optimal amount of worker processes for PHP (throughput vs response time)
- Make sure you can't run out of memory
- Test if your application is stable with APC and turn it on if yes.
- Small changes but they mean 80% increase in performance

Getting better



Configure caching

- Use malloc storage instead of file (no disk IO)
- Define your rules using Varnish VCL
- Set a header to pass original IP to the backend
- FPM and PHP timeouts should match
- Set grace period

Configure caching

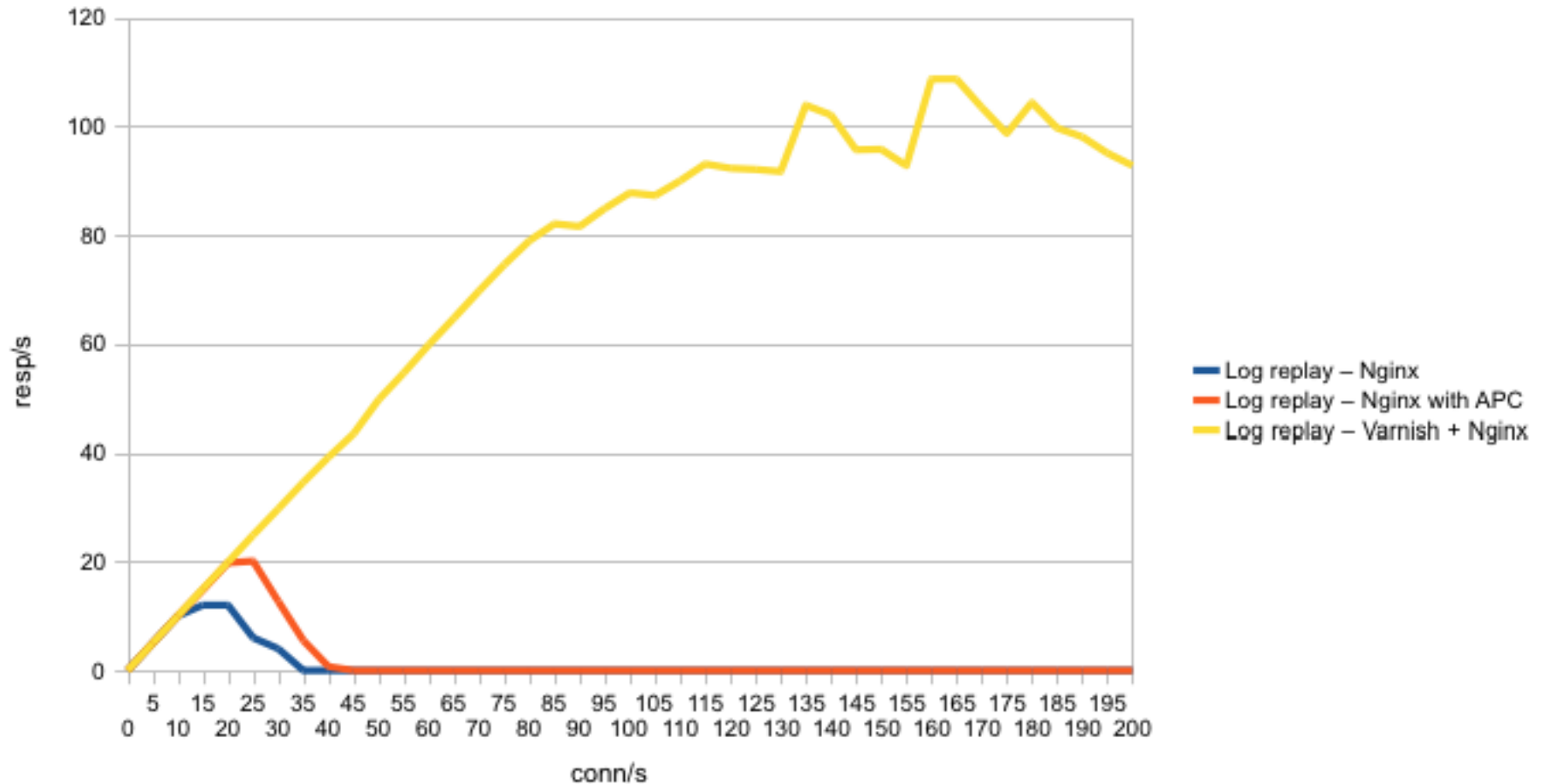
- ```
backend default {
 .host = "127.0.0.1";
 .port = "80";
}
sub vcl_recv {
 set req.backend = default;
 set req.grace = 30s;
 remove req.http.X-Real-IP;
 set req.http.X-Real-IP = client.ip;
 if (req.http.host ~ "test.com(:[0-9]+)?$") {
 if (req.url ~ "nocache") {
 return (pass);
 }
 }
 return (lookup);
}
```

# Real world benchmarking

---

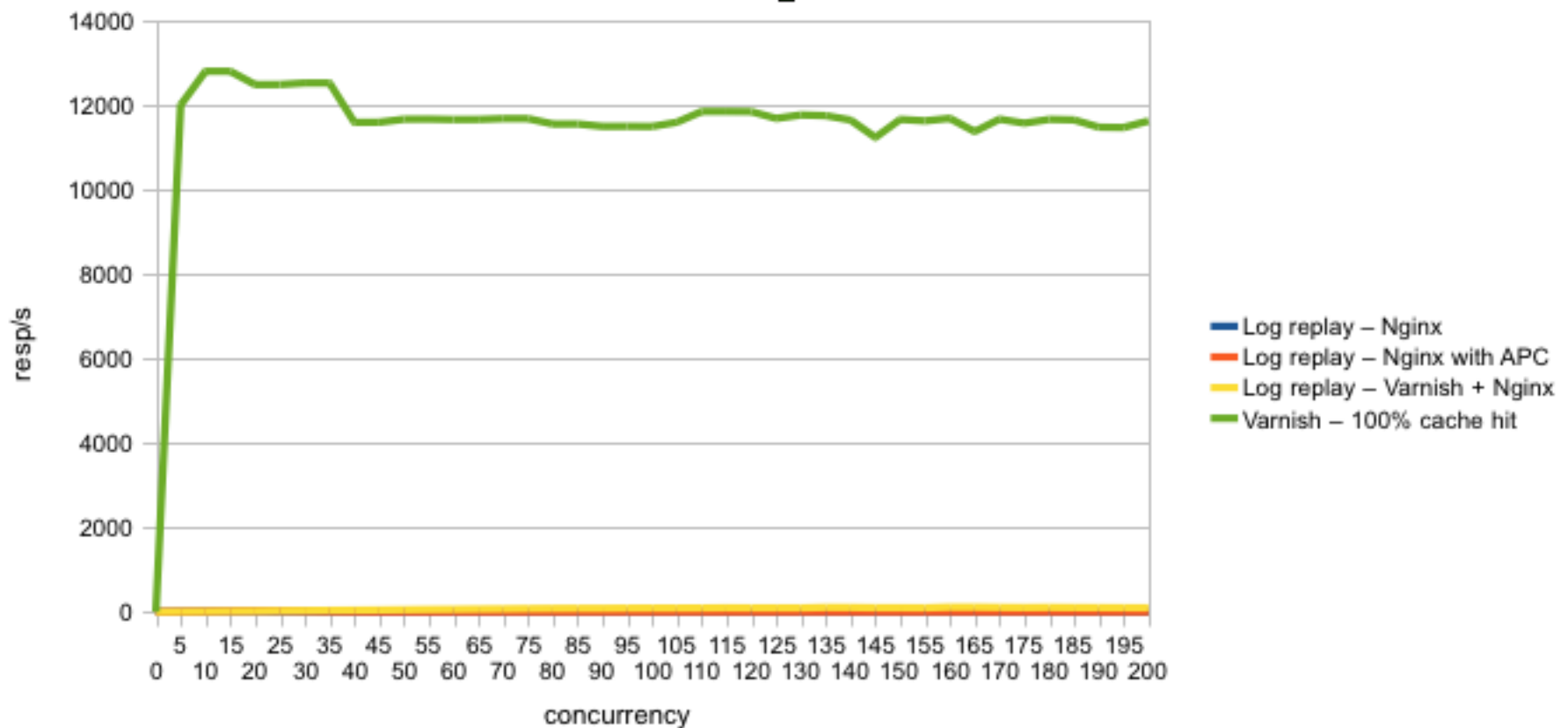
- Necessary as we would get 100% cache hit for a simple benchmark
- Replay last items from access log
- Start with cold cache for each run

# Varnish + Nginx together



# Benchmark individual page

- This is 100% Cache hit scenario



# Benchmarks cont'd

- Cached content loads instantly – no waiting on the user side
- The more (similar) request it receives the faster it becomes – cache hit rate increases
- IO wait and CPU load decreases instantly
- < 3 ms latency
- 5000 req/s under load avg: 1
- Network will be the bottleneck
- But...

# Problems

---

- Session handling
- Logged in users
- Other dynamic parts of the site
- Site content and cache can be out of sync

# Solutions

- Disable cache for logged in users
- Faking dynamic behaviour on cached pages:
  - Edge Side Includes (ESI)
  - Javascript
  - Caching user specific content → different users get different content based on their session id

# Cache invalidation

- Content purged automatically after 5 mins (default)
- Invalidate cached page when content updated
  - Directly via Varnish purge method
  - Pushing requests to a message queue
    - Better for frequently changing content
    - Avoids too many invalidations like user saves content after every change

# Scaling up

---

- Need for more processing power
- Handling different tasks with different servers

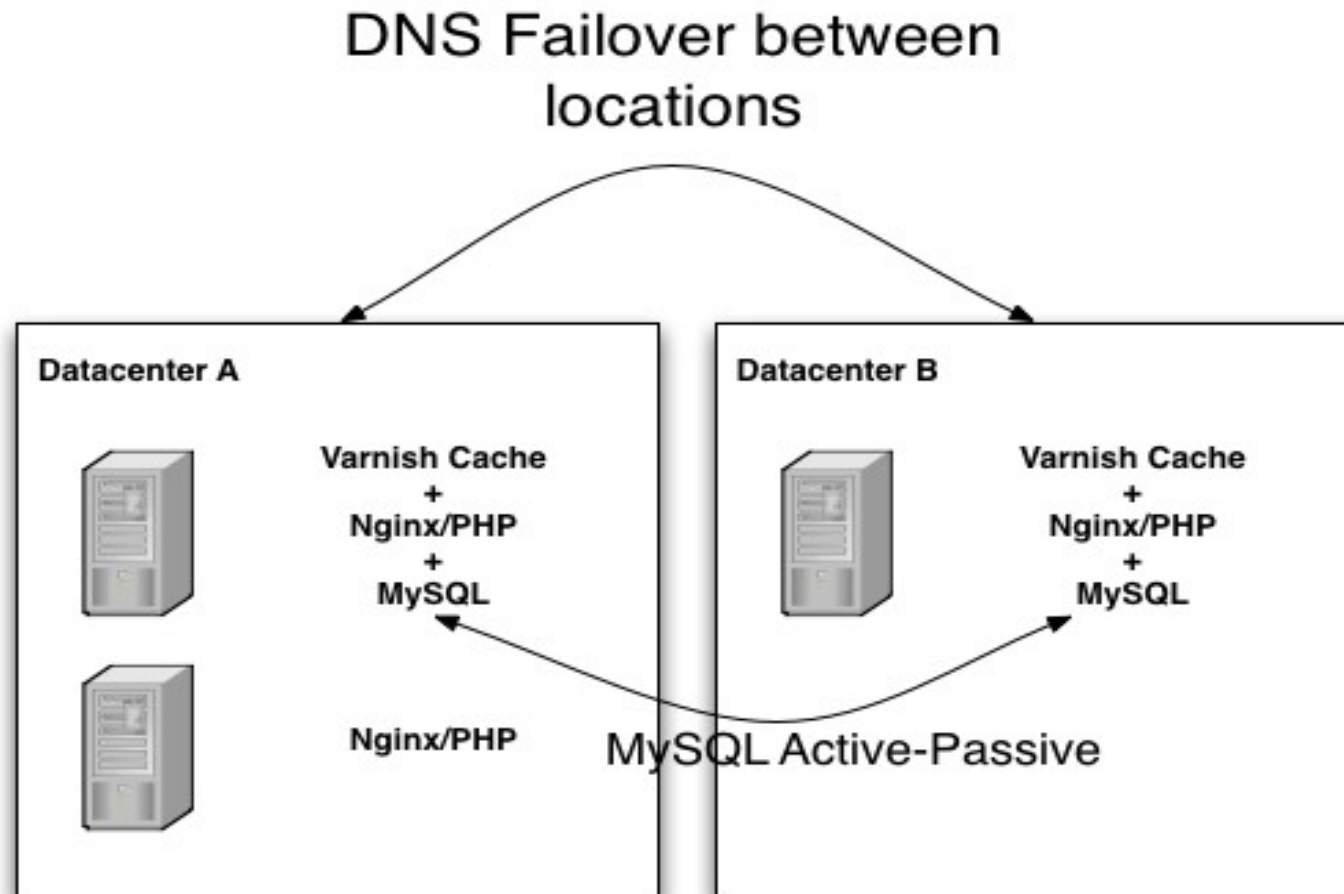
# How?

- Add Nginx/PHP nodes (recommended)

OR

- Add more PHP-FPM nodes
- Balance traffic with Varnish
- Store sessions centrally
- Sync user generated content if needed (rsync, inotify, glusterfs, external storage, application based solution)

# Real world example



Active-Active loadbalance for static content  
Active-Passive mode for dynamic content to avoid double caching

# Results



**Handle the work efficiently with less resources**



# Results

- Enough capacity to serve 6m pageviews / day
- Response time remains stable under extreme load and better response times in general
  - 90-150ms for cache miss
  - Less than 5ms for cached hit
- Users can't notice bad effects of caching
- Server count reduced, less administration, and lower costs

# Conclusions

---

- Flexible and easy to scale up, scale down
- Cache rules can be tricky to write
- Might need to modify app to make it cacheable
- Not just for start-ups where financial resources are limited
- Not for everyone: heavily updated sites need different approach

# Thanks to Our Sponsors

## Platinum Sponsor



## Gold Sponsor



## Silver Sponsors



# Percona Live London Sponsors

## Exhibitor Sponsors



## Friends of Percona Sponsors

**Couchbase**



Monty Program



**Tokutek**



## Media Sponsors



**O'REILLY**

# Annual MySQL Users Conference

## *Presented by Percona Live*

The Hyatt Regency Hotel, Santa Clara, CA

April 10th-12th, 2012

### Featured Speakers

Mark Callaghan, Facebook

Jeremy Zawodny, Craigslist

Marten Mickos, Eucalyptus Systems

Sarah Novotny, Blue Gecko

Peter Zaitsev, Percona

Baron Schwartz, Percona

The Call for Papers is Now Open!

Visit [www.percona.com/live/mysql-conference-2012/](http://www.percona.com/live/mysql-conference-2012/)



**tamas@percona.com**

**We're Hiring! [www.percona.com/about-us/careers/](http://www.percona.com/about-us/careers/)**



[www.percona.com/live](http://www.percona.com/live)