Wix.com

Wix Architecture at Scale

Aviran Mordo
Head of Back-End Engineering @ Wix

 @aviranm

 linkedin.com/in/aviran

 aviransplace.com

Jerry Shaw

/photography portfolio

home nature editorial fashion sport bio

Wix in Numbers

- Over 45,000,000 users
  1M new users/month
- Static storage is >800TB of data
  1.5TB new files/day
- 3 data centers + 2 clouds (Google, Amazon)
  300 servers
- 700M HTTP requests/day
- 600 people work at Wix, of which ~ 200 in R&D

Initial Architecture

Tomcat, Hibernate, custom web framework

- Built for fast development
- Stateful login (Tomcat session), Ehcache, file uploads
- No consideration for performance, scalability and testing
- Intended for short-term use

The Monolithic Giant

- One monolithic server that handled everything
- Dependency between features
- Changes in unrelated areas of the system caused deployment of the whole system
- Failure in unrelated areas will cause system wide downtime

Breaking the System Apart

Concerns and SLA

Edit websites

- Data Validation
- Security / Authentication
- Data consistency
- Lots of data

View sites, created by Wix editor

- High availability
- High performance
- High traffic volume
- Long tail

Serving Media

- High availability
- High performance
- Lots of static files
- Very high traffic volume
- Viewport optimization
- Cacheable data

Wix Segmentation

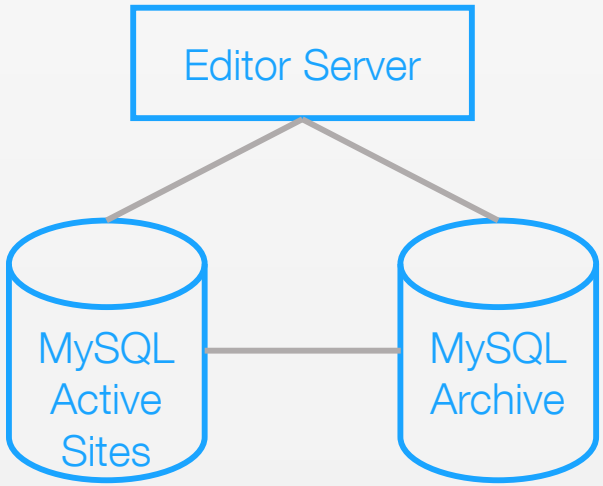
Making SOA Guidelines

- Each service has **its own** database (if one is needed)
- Only **one** service can write to a specific DB
- There may be additional **read-only** services that directly accesses the DB (for performance reasons)
- Services are **stateless**
- No DB transactions
- Cache is **not** a building block, **but an optimization**

1. Editor Segment

Editor Server

- Immutable JSON pages (~2.5M / day)
- Site revisions
- Active – standby MySQL cross datacenters

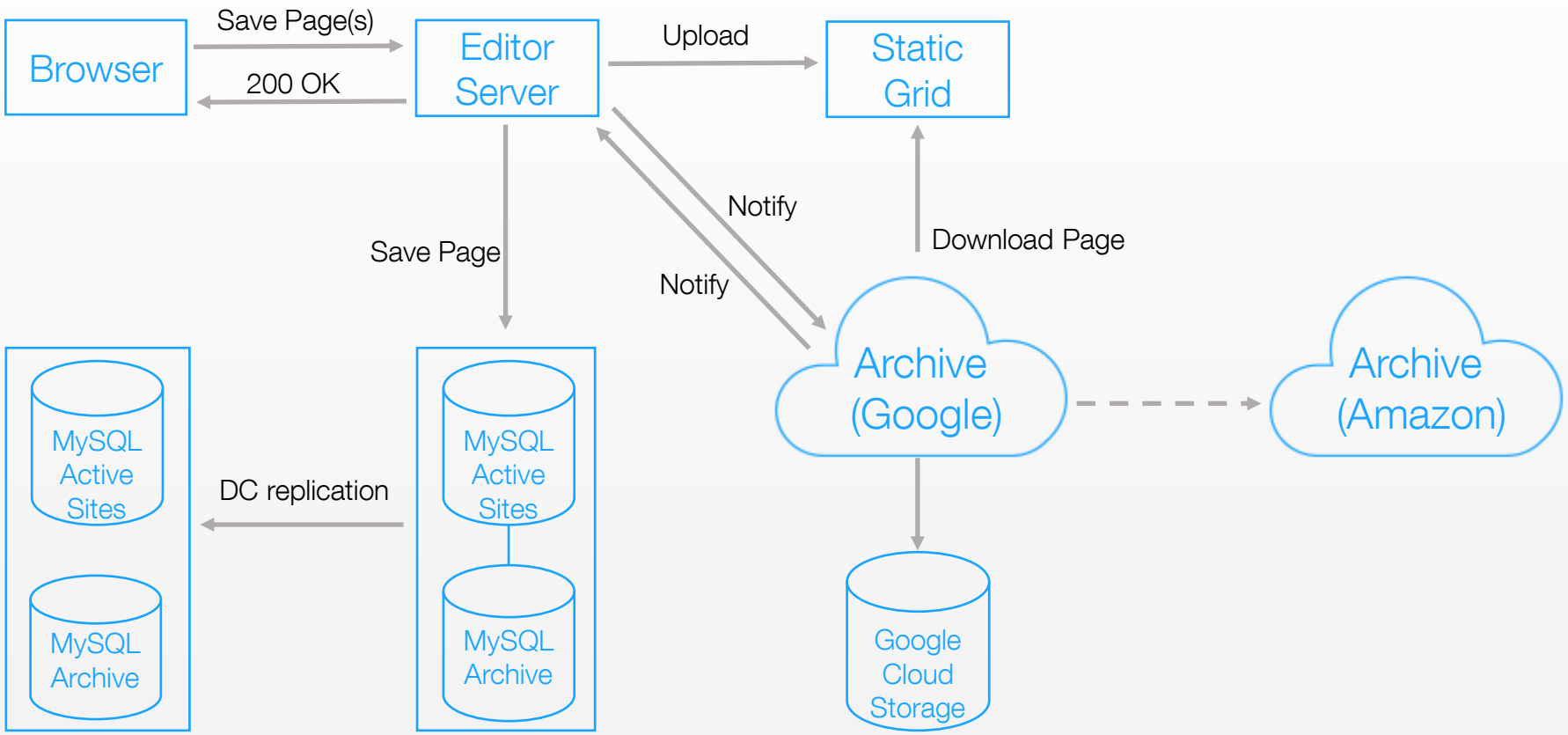



Find Your Critical Path
PREPARE FOR
THE END OF
THIS WORLD

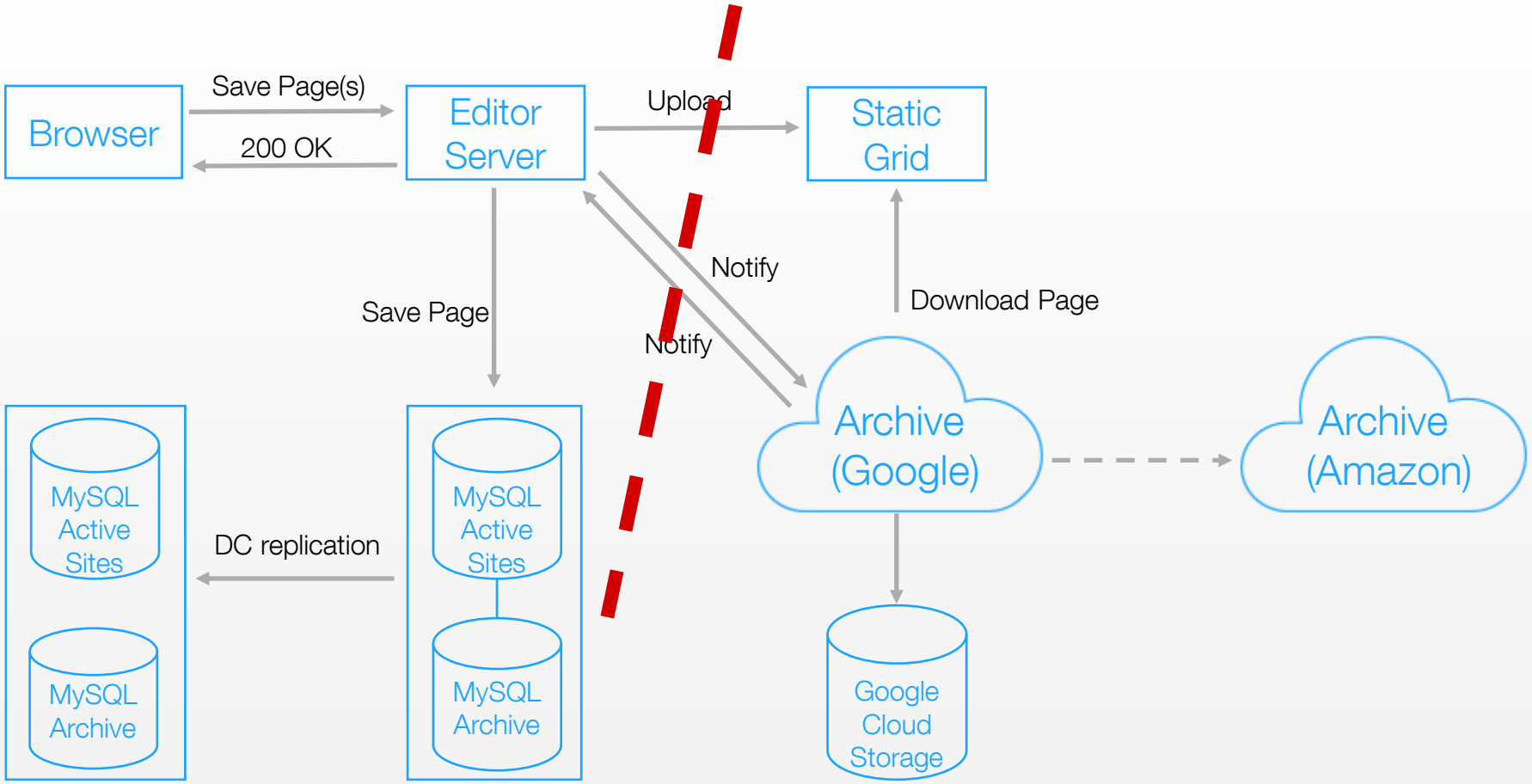
Protect The Data

- Protect against DB outage with fast recovery = replication
- Protect against data poisoning/corruption = revisions / backup
- Make the data available at all times = data distribution to multiple locations / providers

Saving Editor Data



Self Healing Process



No DB Transactions

- Save each page (JSON) as an atomic operation
- Page ID is a content based hash (immutable/idempotent)
- Finalize transaction by sending site header (list of pages)
- Can generate orphaned pages, not a problem in practice

2. Media Segment

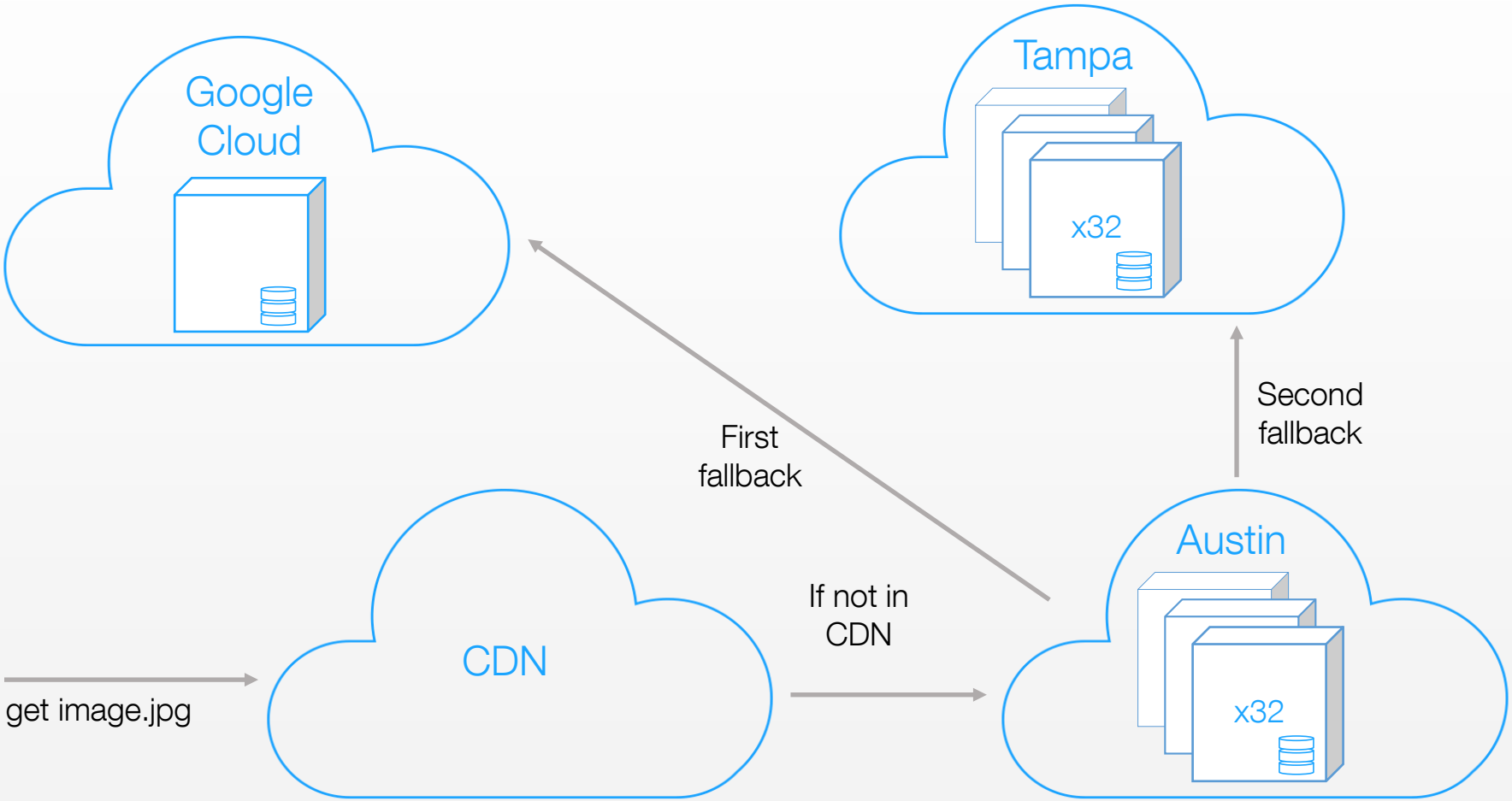
Prospero – Wix Media Storage

- 800TB user media files
- 3M files uploaded daily
- 500M metadata records
- Dynamic media processing
  - Picture resize, crop and sharpen “on the fly”
  - Watermark
  - Audio format conversion

Prospero

- Eventual consistent distributed file system
- Multi datacenter aware
- Automatic fallback cross DC
- Run on commodity servers & cloud

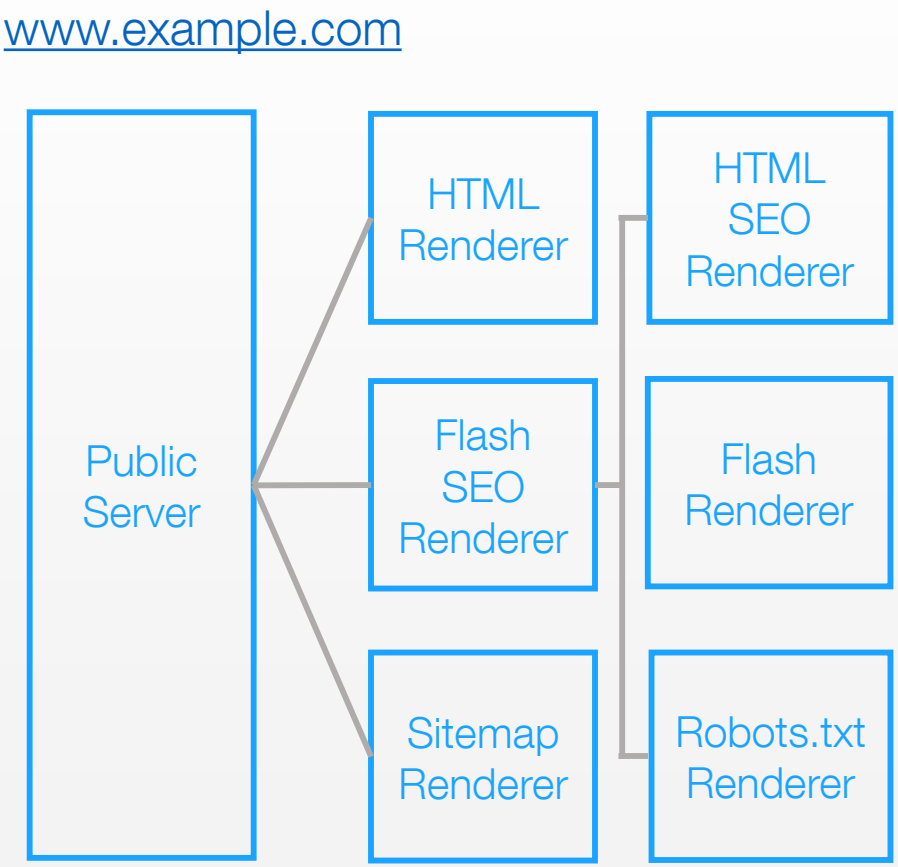
Prospero – Wix Media Manager



3. Public Segment

Public Segment Roles

- Routing (resolve URLs)
- Dispatching (to a renderer)
- Rendering (HTML,XML,TXT)

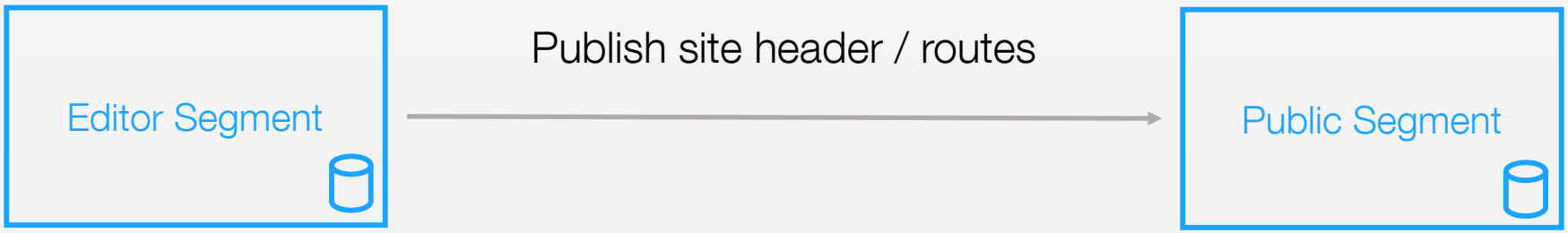


Public SLA

Response time <100ms at peak traffic

Publish A Site

- Publish site header (a map of pages for a site)
- Publish routing table



Built For Speed

Minimize out-of-service hops (2 DB, 1 RPC)

Lookup tables are cached in memory, updated every 5 minutes

Denormalized data – optimize for read by primary key (MySQL)

Minimize business logic

How a Page Gets Rendered

Bootstrap HTML template that contains only data

- Only JavaScript imports
- JSON data (site-header + dynamic data)
- No “real” HTML view

Offload rendering work to the browser

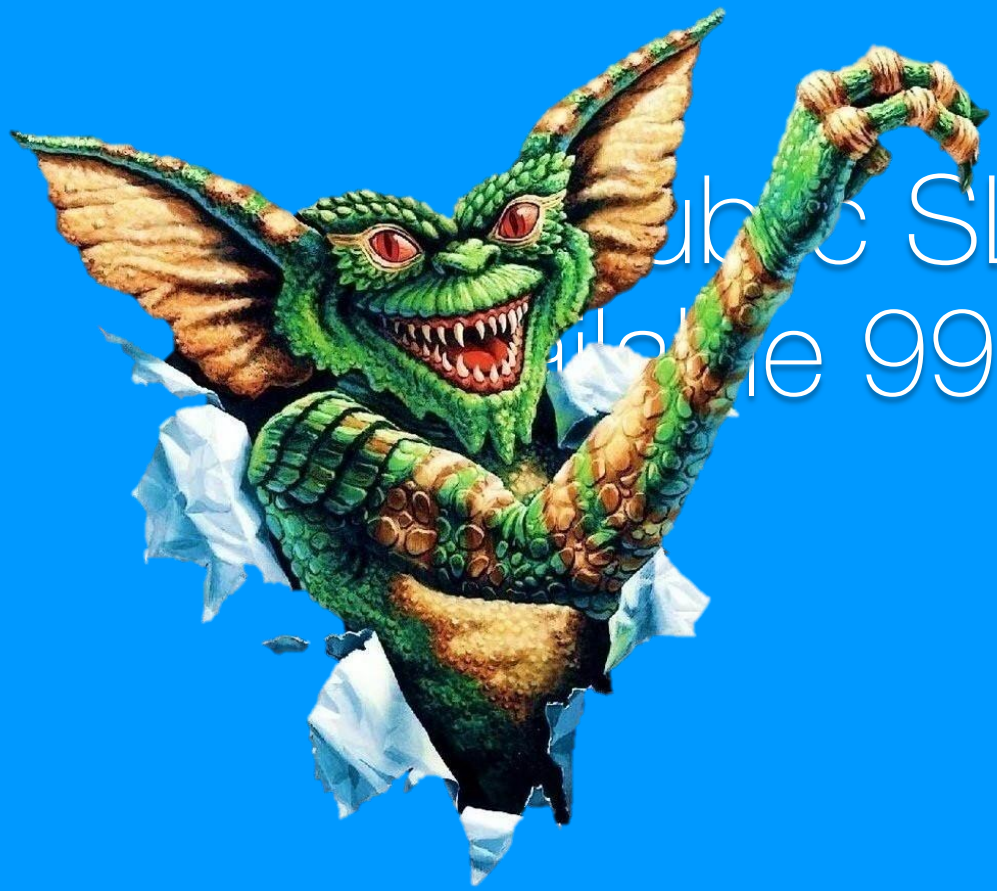
The average Intel Core i750 can push up to 7 GFLOPS without overclocking

Why JSON?

- Easy to parse in JavaScript and Java/Scala
- Fairly compact text format
- Highly compressible (5:1 even for small payloads)
- Easy to fix rendering bugs (just deploy a new client code)

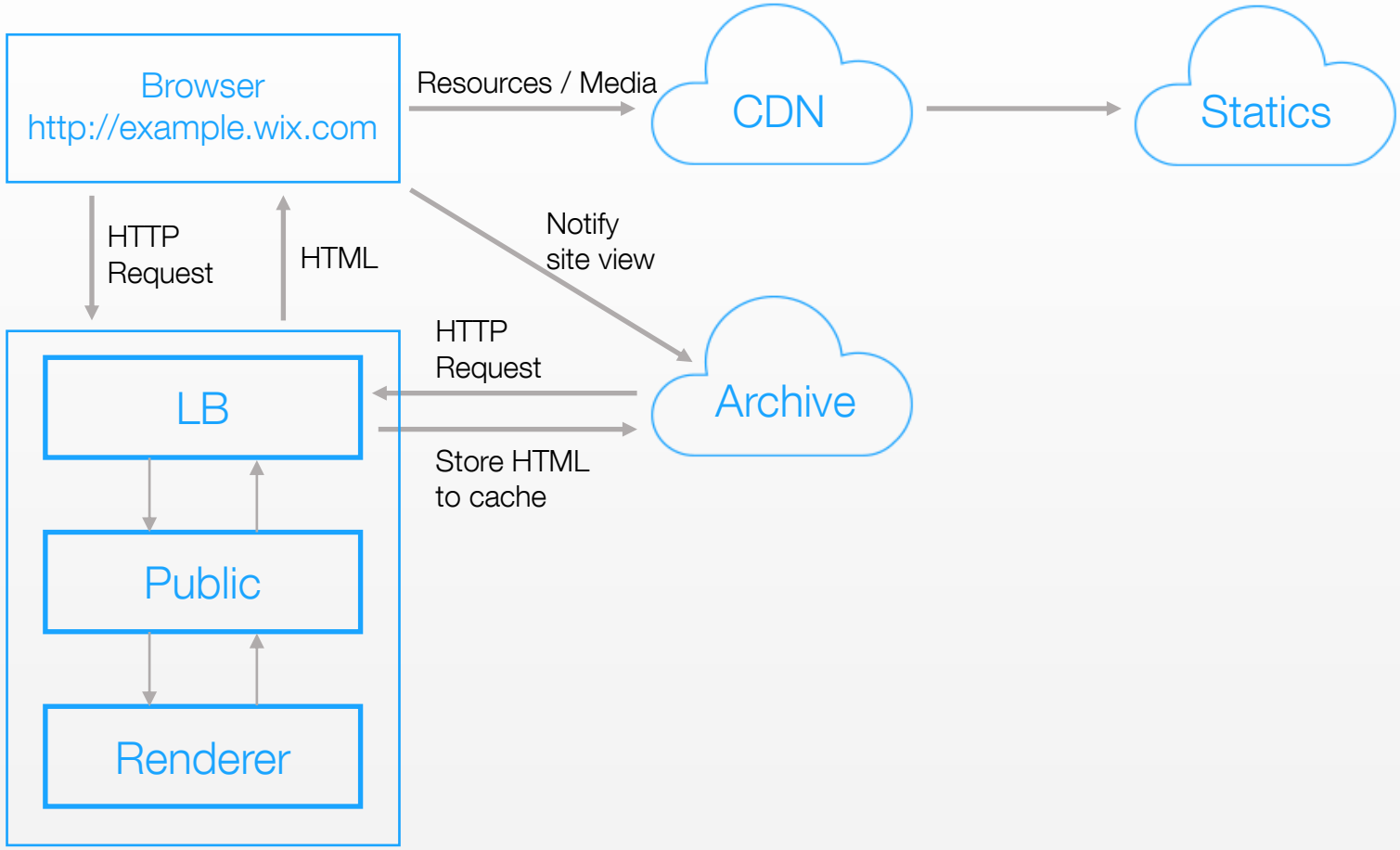
Minimum Number of Public Servers Needed to Serve 45M Sites

4

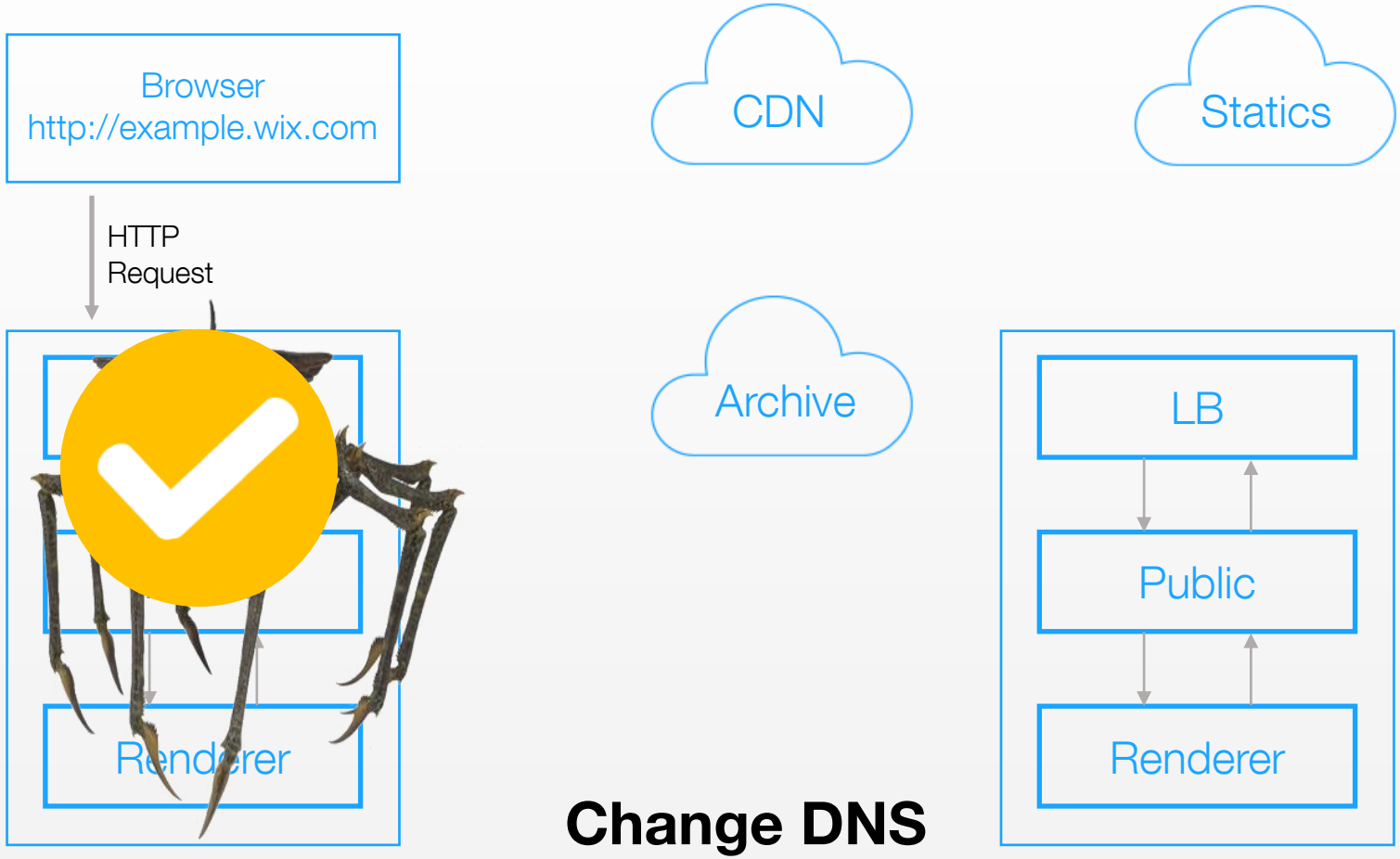


ub c SLA
ilable 99.99999%

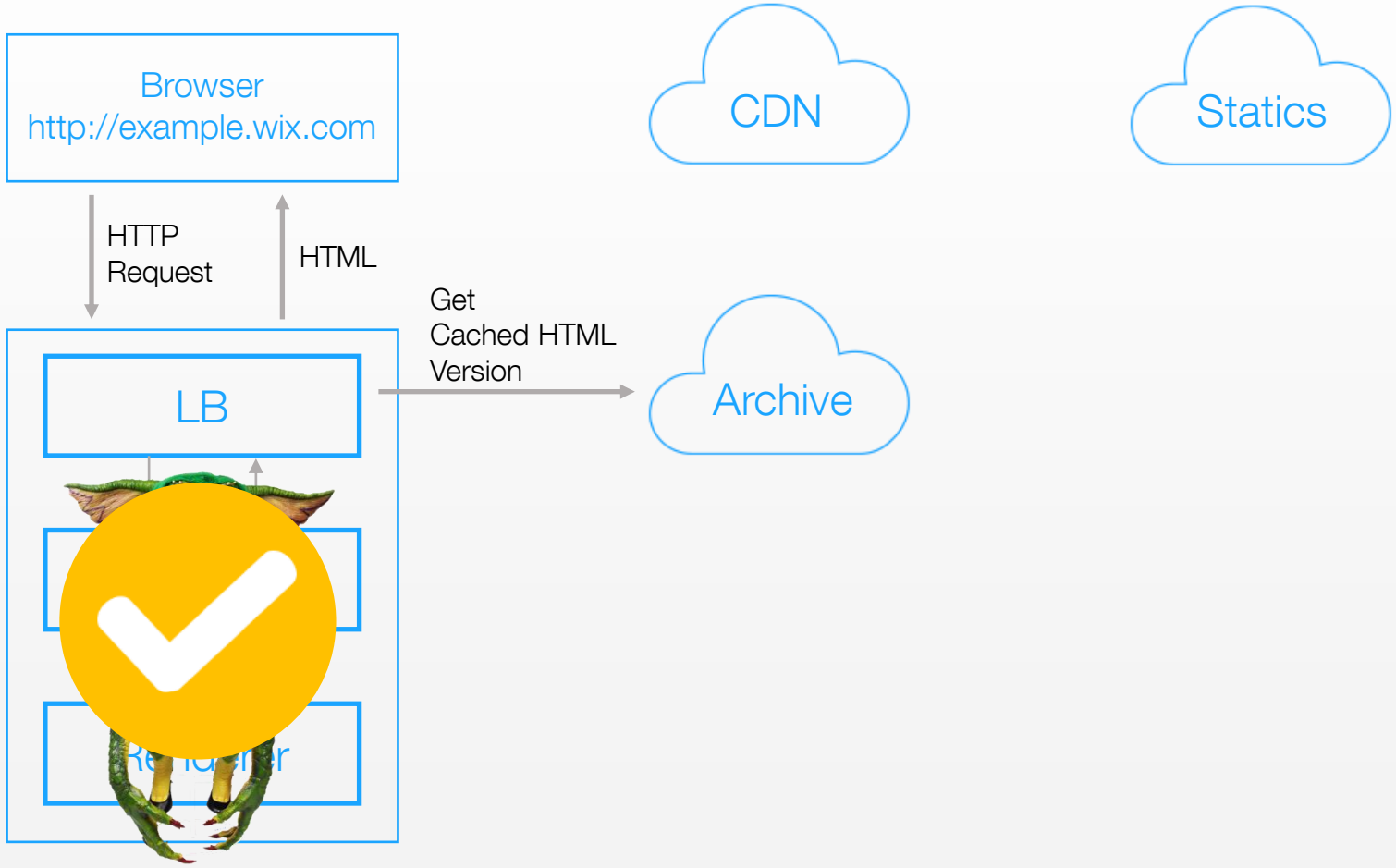
Serving a Site – Sunny Day



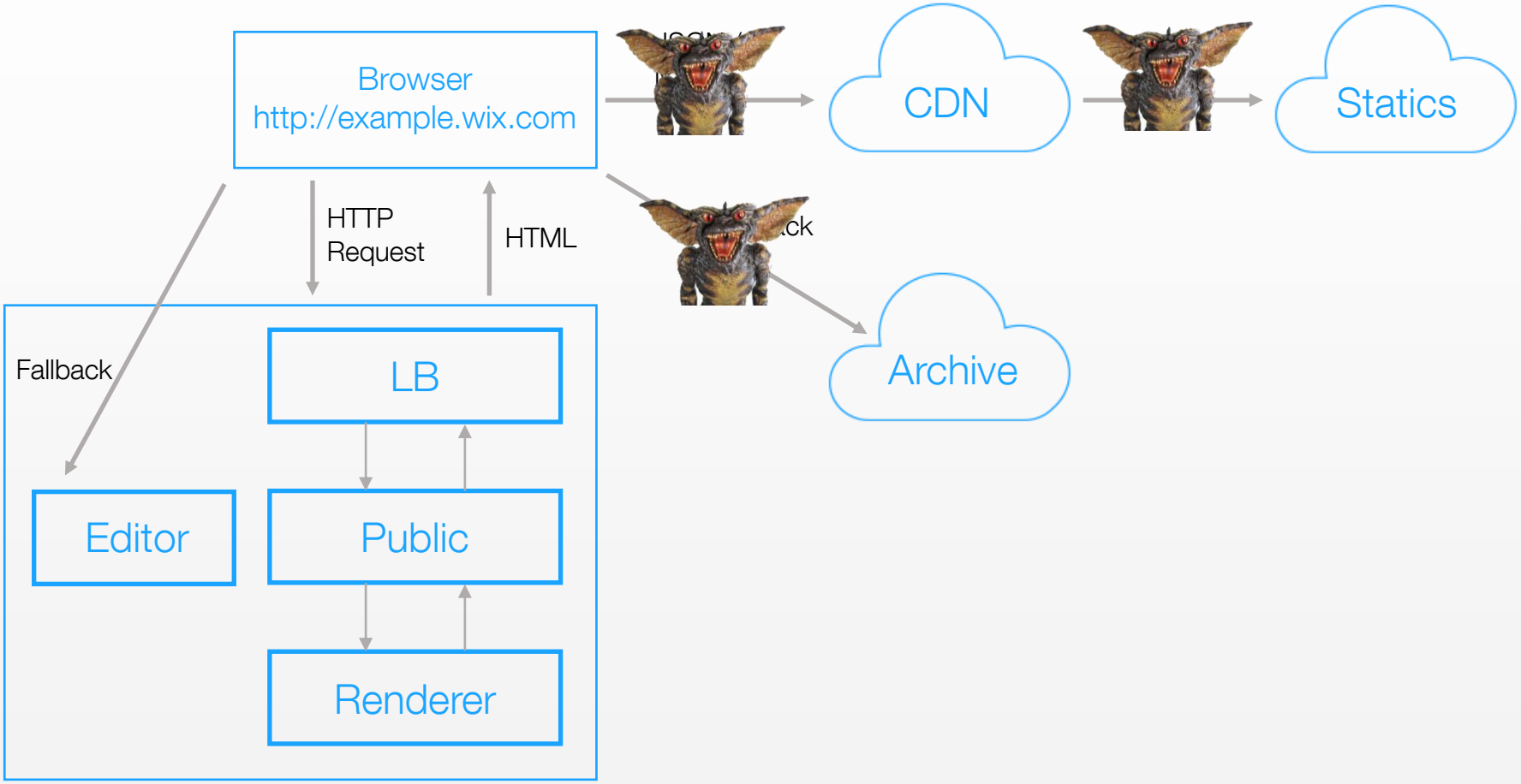
Serving a Site – DC Lost



Serving a Site – Public Lost



Living in the Browser



Summary

- Identify your critical path and concerns
- Build redundancy in critical path (for availability)
- De-normalize data (for performance)
- Minimize out-of-process hops (for performance)
- Take advantage of client's CPU power

Q&A

 http://goo.gl/Oo3lGr

Aviran Mordo
Head of Back-End Engineering @ Wix

 @aviranm

 linkedin.com/in/aviran

 aviransplace.com

QCon

Please evaluate my talk via the mobile app!