web performance Barcelona

Image Optimisation for WebApps

Nil Portugués Caldero

Existing solutions

Used either during **dev** or **production deploy** stage:

- **Symfony2:** Assetic Bundle
- **NodeJS:** github.com/gruntjs/grunt-contrib-imagemin
- **Ruby/RoR:** github.com/toy/image_optim

Existing solution problems

- Not a real-time solution.
- Waiting for images does not scale. Slow deployment.
- **No strategy!**

Existing requirements

Device diversity: phone, tablet, PC require a strategy.

- Different presentation layers (media queries)**. Needs Strategy.**
- Different design requirements**. Needs Strategy.**
- Fast multi-device UX**. Needs Strategy.**

Strategy: eventual consistency

Is a **consistency model** used in **distributed computing** to achieve **high availability** that informally guarantees that, if no new updates are made to a given data item, eventually all accesses to that item will return the last updated value.

Source: *Wikipedia*

Solution: Conceptual

Model

Extract and fetch optimized images (if any) for the current View

Controller

Use the best available image.

Image Service

Send image to optimise to Queue (if image is not in database)

View

Renders HTML

Background process:

Queue with Image Optimisation workers

Model: Add abstraction layer

- Separate image assets from the application by extracting.
- Replace image asset for references. Best way, md5 file contents hash. This will be our **image token**.
- Store image references and relevant data in a persistence layer (eg: SQL table or MongoDB document)

Model: Add abstraction layer

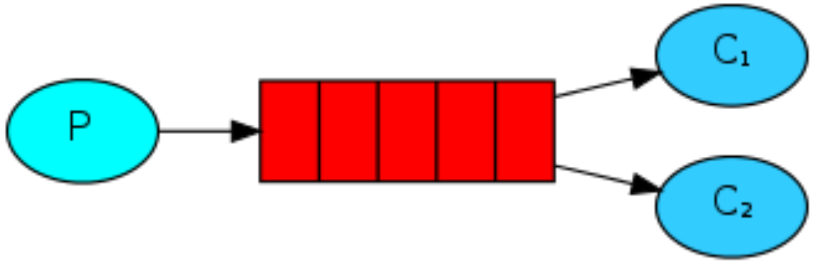
```
<!-- $html will be transformed to $processedHtml -->
<img src="http://example.com/path/to/image/image.jpg" style="
border:2px solid red" data-attribute="example1">
```

```
<!-- $processedHtml -->
{{IMG|6fd86da74659f04253285e853af26845|style="border:2px solid
red"|data-attribute="example1"}}
```

Image Service: Send to queue

- Use queues to **resize** images (optional) and **optimise** them.

- Queue system have drivers for many languages.
  - RabbitMQ
  - HornetQ
  - Apache ActiveMQ
  - Apache Apollo
  - Apache Qpid
  - ...



Controller: Use best image available

Fetch the best image possible using the reference:

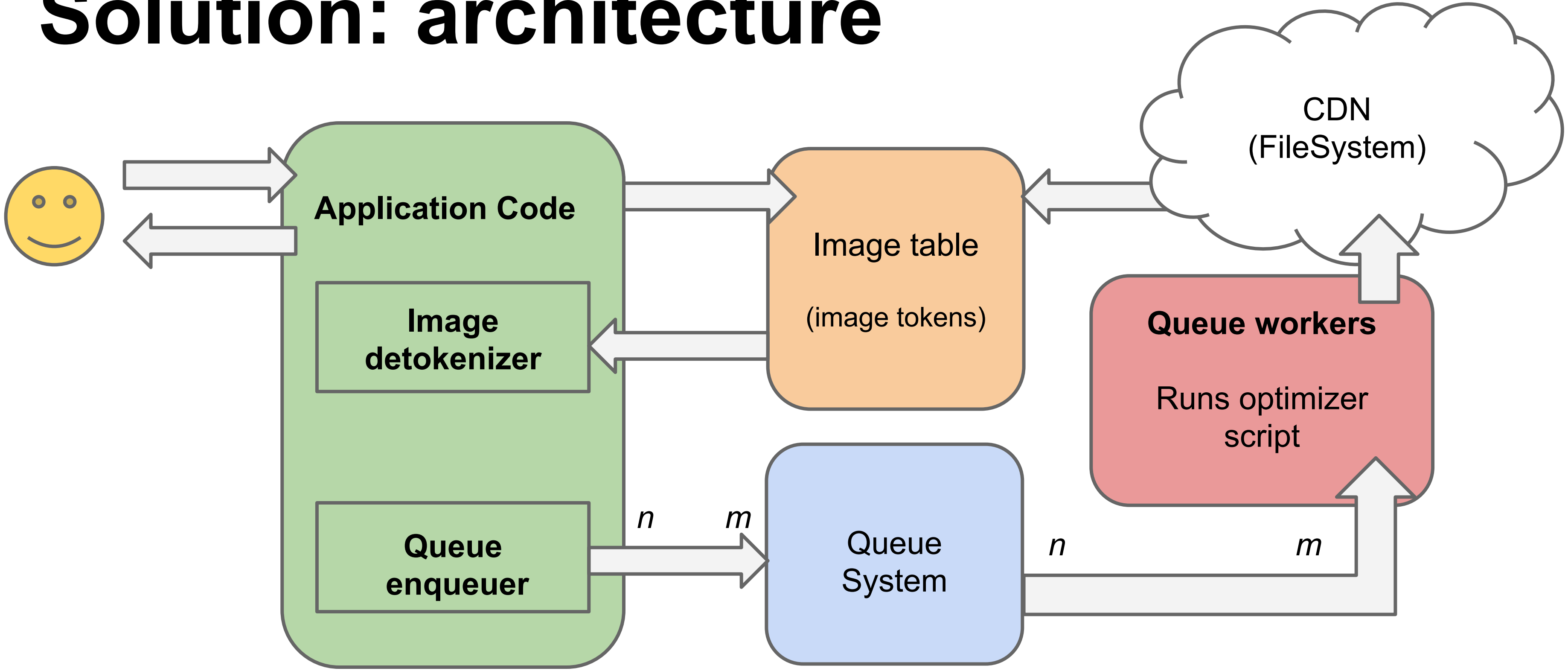
- ○ If image has not been processed, use original.
- ○ If image has been processed, used optimised version.

Replace **image token** with the **most performant** image for the **current browser and device.**

Queue workers

- The more queue workers, the faster images will be processed.
- Priority queues can be set depending on criteria to speed up.
- CPU usage for these machines will be high.
- These machines should be able to write to a CDN server.

Solution: architecture



Consistency model + high availability + distributed computing

Optimiser Strategy: The Tools

- Image optimisation tools are very specialised.
- Yet they are dumb, cannot work together.
- So let’s fix it and make them play together, as one.

Optimiser Strategy: The Tools

- **JPG:** jpegoptim, libjpeg-progs
- **PNG:** pngout, pngtools, advpng, pngcrush, optipng, pngquant
- **GIF:** gifsicle
- **SVG:** SVGCleaner
- **WebP:** Google’s image format + encoder.

Optimiser Strategy: Big Picture

- Changing file format (eg: PNG to JPG, JPG to WEBP, GIF to PNG) is absolutely OK.
- Lossless conversion is default, except for PNG to JPG.
- Run image optimizers in parallel...
- … so we can compare output file size and keep the smallest.

Optimiser Strategy: JPG

- Convert input image to have both progressive and baseline versions JPGs.

- Remove EXIF and meta-data.

- Do JPG to WEBP conversion for supported browsers

- Compare output and keep the smallest JPG and WEBP if smaller than produced JPG.

Optimiser Strategy: PNG

- Try reduce channels, from 32 to 24 to 8 bits when possible, depending on the image colors.

- Alpha files 32 bits usually works as 24 bits

- Re-compress PNG with an optimized LWZ compressor.

Optimiser Strategy: GIF

- Try re-compression.
- If single-framed GIF, go for a PNG, usually produces smaller file sizes.
- For animated GIF, just try frame optimization. No PNG.

Optimiser Strategy: SVG

- Clean it up using SVG Cleaner client.

- Compare output and keep the smallest.

Optimiser Strategy: The worker

- Runs the script. Returns a JSON straight from command-line to worker.
- Worker reads the JSON and picks up the best image.
- Upload to CDN.
- Adds to the database the optimised version/s reference plus additional data such as mime-type.

web performance Barcelona

Questions?