The Cache Delusion

Not a Golden Hammer

TOBIAS
SCHLITT

Hello, I’m Toby!

Principal Engineer (backend) at

commercetools
Frontend

You might also know me from:

FRONTASTIC

tideways

Qafoo
passion for software quality

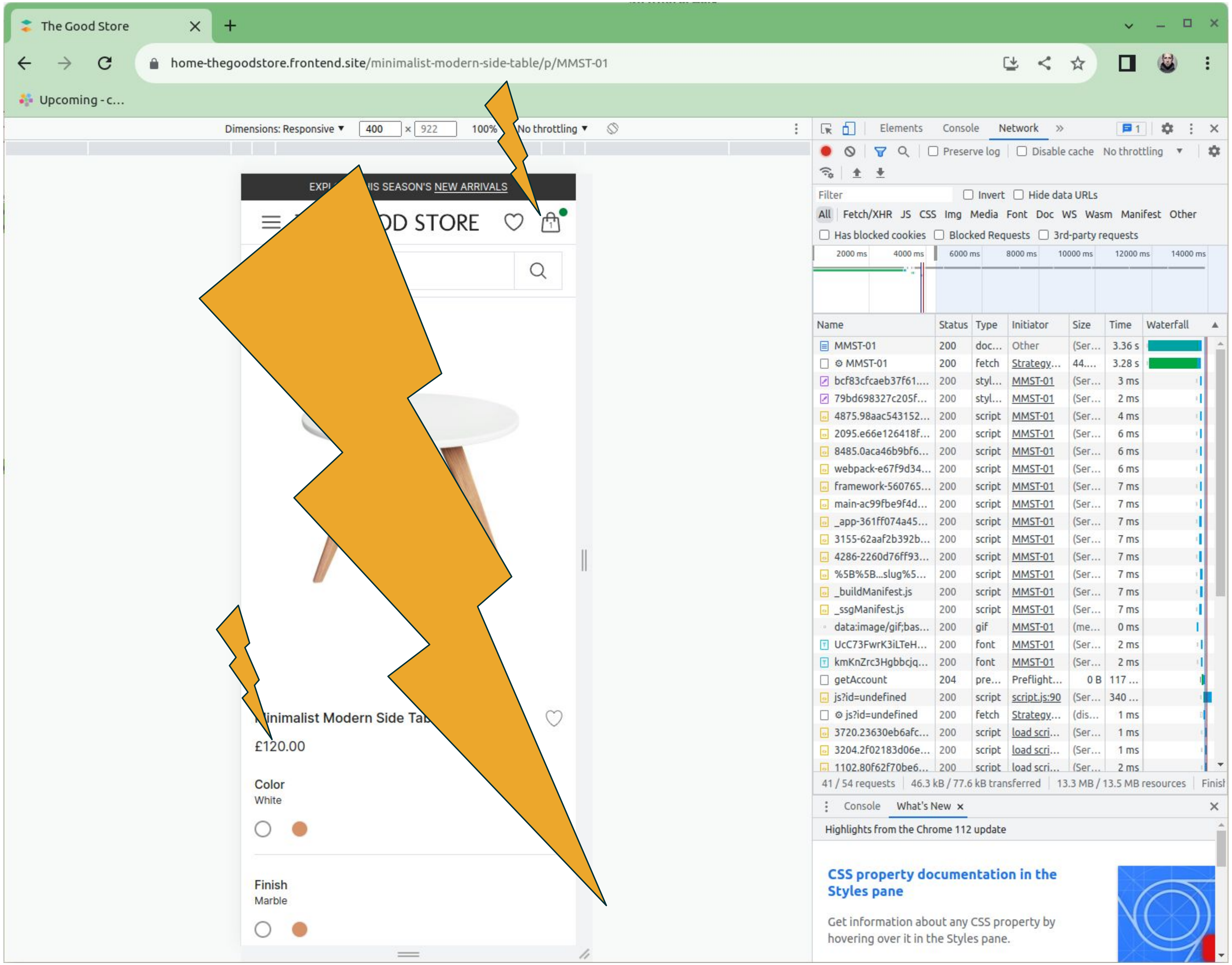
Why this talk?

MY SQL QUERY IS SLOW
CACHING!
imgflip.com

I GET TOO MUCH TRAFFIC
CACHING
imgflip.com

MY API VENDOR IS SLOW
CACHING
imgflip.com

MY PAGE LOADS SLOW
CACHING!!1!
imgflip.com



home-thegoodstore.frontend.site/minimalist-modern-side-table/p/MMST-01

Dimensions: Responsive ▾ 400 × 922 100% ▾ No throttling ▾

EXPL… IS SEASON'S NEW ARRIVALS

…OD STORE

…inimalist Modern Side Tab…

£120.00

Color
White

○ ●

Finish
Marble

○ ●

Name	Status	Type	Initiator	Size	Time
MMST-01	200	doc…	Other	(Ser…	3.36 s
⚙ MMST-01	200	fetch	Strategy…	44.…	3.28 s
bcf83cfcaeb37f61.…	200	styl…	MMST-01	(Ser…	3 ms
79bd698327c205f…	200	styl…	MMST-01	(Ser…	2 ms
4875.98aac543152…	200	script	MMST-01	(Ser…	4 ms
2095.e66e126418f…	200	script	MMST-01	(Ser…	6 ms
8485.0aca46b9bf6…	200	script	MMST-01	(Ser…	6 ms
webpack-e67f9d34…	200	script	MMST-01	(Ser…	6 ms
framework-560765…	200	script	MMST-01	(Ser…	7 ms
main-ac99fbe9f4d…	200	script	MMST-01	(Ser…	7 ms
_app-361ff074a45…	200	script	MMST-01	(Ser…	7 ms
3155-62aaf2b392b…	200	script	MMST-01	(Ser…	7 ms
4286-2260d76ff93…	200	script	MMST-01	(Ser…	7 ms
%5B%5B...slug%5…	200	script	MMST-01	(Ser…	7 ms
_buildManifest.js	200	script	MMST-01	(Ser…	7 ms
_ssgManifest.js	200	script	MMST-01	(Ser…	7 ms
data:image/gif;bas…	200	gif	MMST-01	(me…	0 ms
UcC73FwrK3iLTeH…	200	font	MMST-01	(Ser…	2 ms
kmKnZrc3Hgbbcjq…	200	font	MMST-01	(Ser…	2 ms
getAccount	204	pre…	Preflight…	0 B	117 …
js?id=undefined	200	script	script.js:90	(Ser…	340 …
⚙ js?id=undefined	200	fetch	Strategy…	(dis…	1 ms
3720.23630eb6afc…	200	script	load scri…	(Ser…	1 ms
3204.2f02183d06e…	200	script	load scri…	(Ser…	1 ms
1102.80f62f70be6…	200	script	load scri…	(Ser…	2 ms

41 / 54 requests | 46.3 kB / 77.6 kB transferred | 13.3 MB / 13.5 MB resources | Finish

Highlights from the Chrome 112 update

CSS property documentation in the Styles pane

Get information about any CSS property by hovering over it in the Styles pane.

What actually is a cache?

```
function someFunction()
{
    return timeConsumingOperation();
}
```

```
function someFunction()
{
    $result = $cache->get(‘time-consuming-operation-result’);
    if (!$result) {
        $result = timeConsumingOperation();
        $cache->put(‘time-consuming-operation-result’, $result);
    }
    return $result;
}
```

Variations

- Execute `timeConsumingOperation()` in another process
    - Re-calculation does not happen on the fly
- Implement cache for a larger number of operations in a dedicated layer
    - For example in an event system / pipes & filters
- Various individual adjustments …

Goals of this talk

- Many might know: I try to avoid caching where possible

- This talk should give you some insights into: WHY?

- It also should give you some tools to evaluate
  when and how
  caching can be a solution

Factors of cache design

01 Cache layer level

02 Currentness expectations

03 Cache dimensions

04 Purging / invalidation strategy

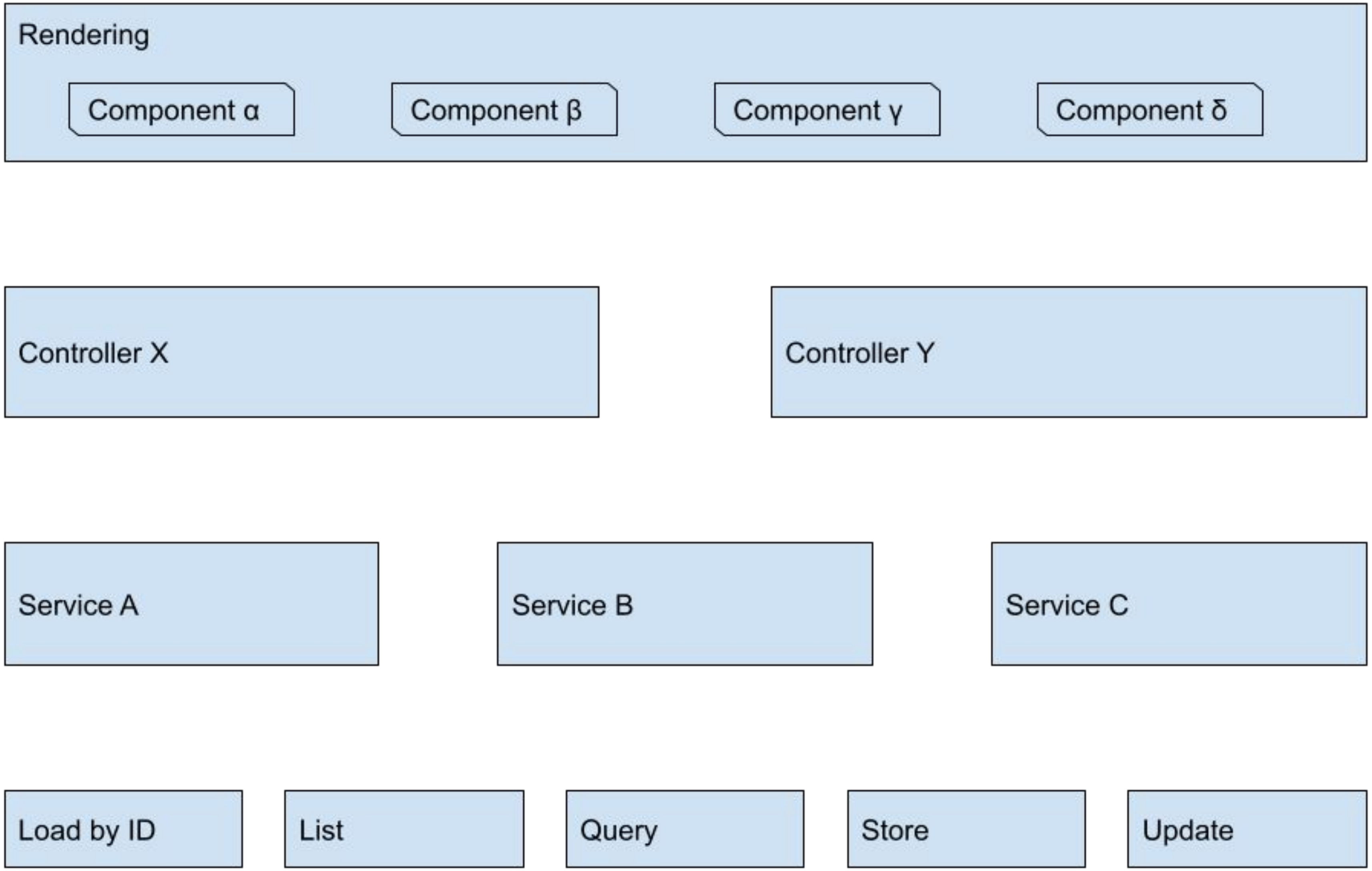
05 Examples: Caching gone wrong

06 Lessons to be learned

Caching on layers

… or where to put the pitfalls

TOBIAS
SCHLITT



Rendering

Component α

Component β

Component γ

Component δ

Controller X

Controller Y

Service A

Service B

Service C

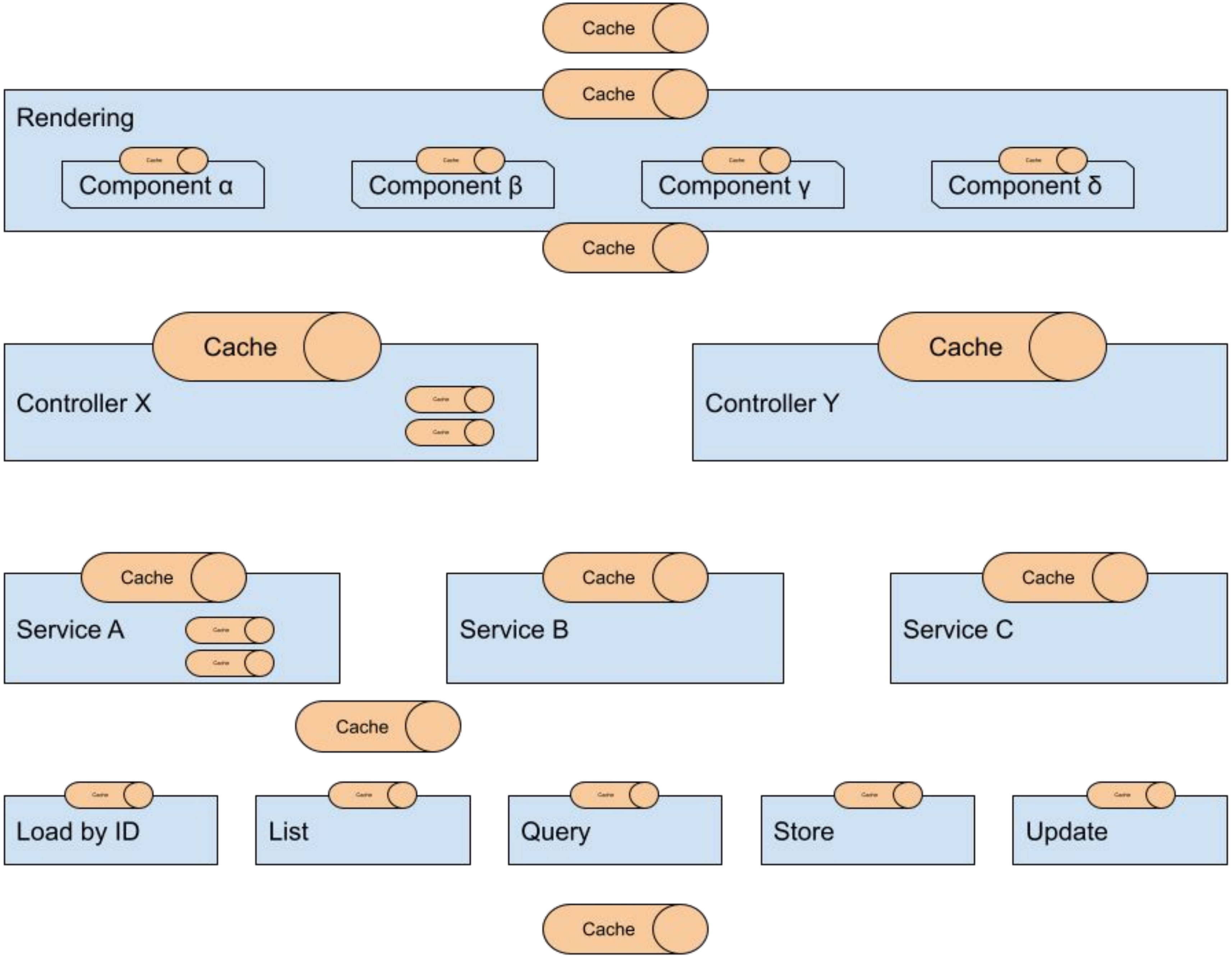
Load by ID

List

Query

Store

Update



Cache

Cache

Rendering

Cache Cache Cache Cache

Component α

Component β

Component γ

Component δ

Cache

Cache

Controller X

Cache

Cache

Cache

Controller Y

Cache

Service A

Cache

Cache

Cache

Service B

Cache

Service C

Cache

Cache Cache Cache Cache Cache

Load by ID

List

Query

Store

Update

Cache

Select the layer for caching wisely

- Choose only a single layer for the cache
- Never implement caches on multi-level!
- If possible: Put the cache outside of the system
  - e.g. full-page frontend caching
  - e.g. HTTP cache in front of a REST API

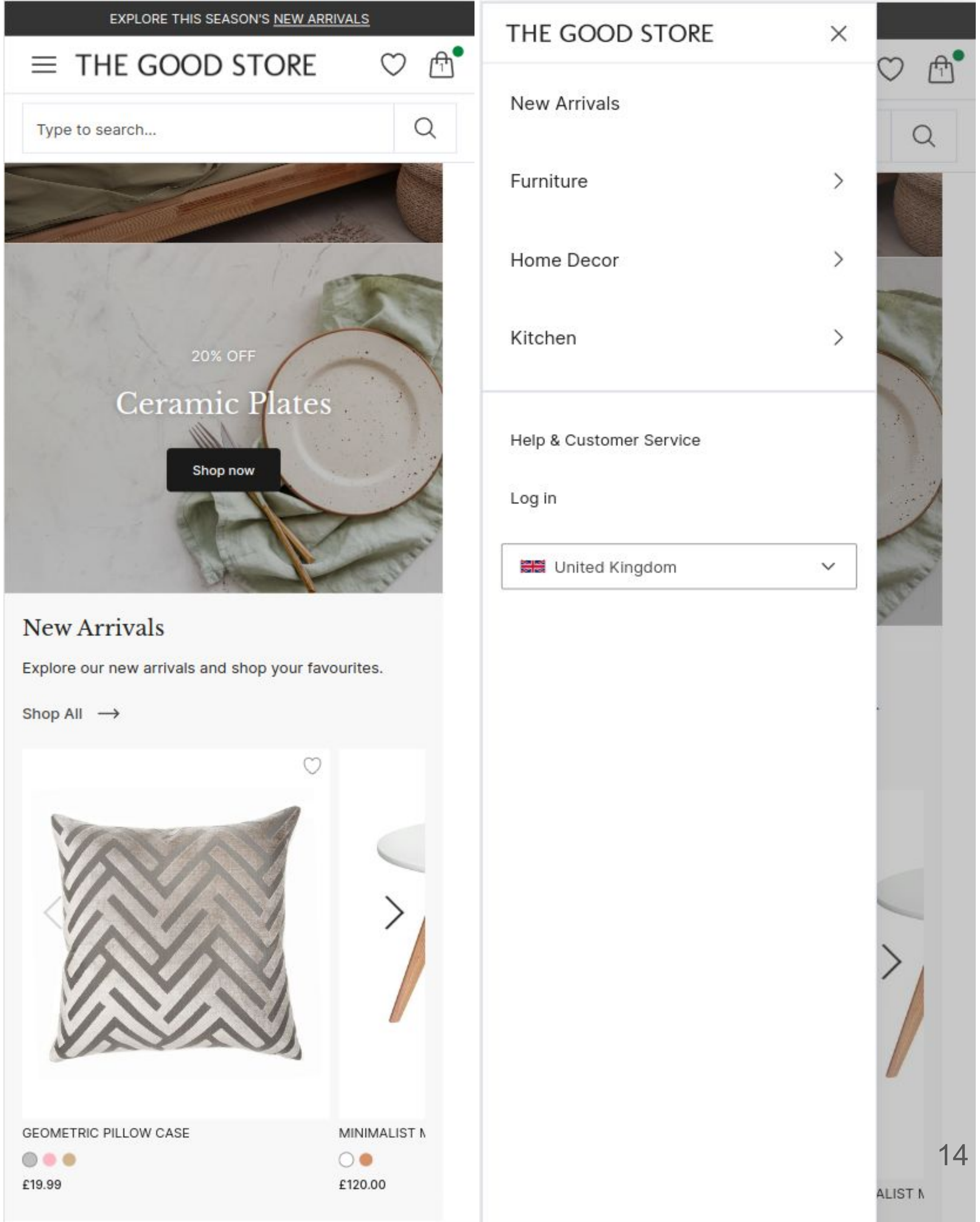
TOBIAS SCHLITT

Currentnes expectations

… or why caching is a business decision

Currentness expectations

- How long would you expect this page can be cached?



Explore our new arrivals and shop your favourites.

Currentness expectations

- Typical cache times by example:
  - 1 day (sitemap)
  - 15 minutes (start page with news articles)
  - 1 second (stock price on trading platform)
- If you cache complex structures: What is the most time-critical part?

Currentness expectations

- Do you accept staleness during re-calculation?
- Do you accept temporary inconsistencies?

Cache dimensions

… or how a cache becomes a system component

TOBIAS
SCHLITT

What cache size do you need?

- How big is a data item you want to cache?
  - 16 KB single coco product as JSON
  - 250 KB homepage, HTML only
  - 1.9 MB compiled Symfony container as PHP files
- How many items to you need to cache?
  - 75k products in a shop
  - 30k pages in a site (per language)
  - 1 compiled Symfony container per server
- + Meta-Data overhead, index, …

Where & how fast do you need to access your cache?

● CPU L1 cache	0.5 ns	Local server!
● 1 MB in RAM	250 µs	
● 1 MB on SSD	1 ms	
● Network roundtrip CA -> NL -> CA	150 ms	
● + Processing overhead in your program		

(1 ms = 1,000 µs = 1,000,000 ns)

- Credits: Latency numbers every programmer should know

Typical cache storages

- Local hard disk / SSD
- Memcache
- Redis
- *MySQL

Purging / invalidation

… or when complexity kicks in

Invalidation or purging

- 2 typical invalidation strategies:
  - Passive invalidation via TTL (time to live)
  - Active via purge-on-update
- Let’s see the complexity in the upcoming examples

If your cache is too small

- Fast caches are typically small
- Cache purging strategies need to kick in, choose a strategy:
  - FIFO (first in, first out)
  - LRU (least recently used)
  - LFU (least frequently used)
  - Want more? Find here: https://en.wikipedia.org/wiki/Cache_replacement_policies

Examples: Caching gone wrong

… or how I shot myself in the foot, Toby-edition

TOBIAS
SCHLITT

Login: The naive full page cache

- Idea:
  - Cache all pages fully for 120 seconds
- Pitfall:
  - Expose logged in user data to next visitor

CMS: Purge when page data changes

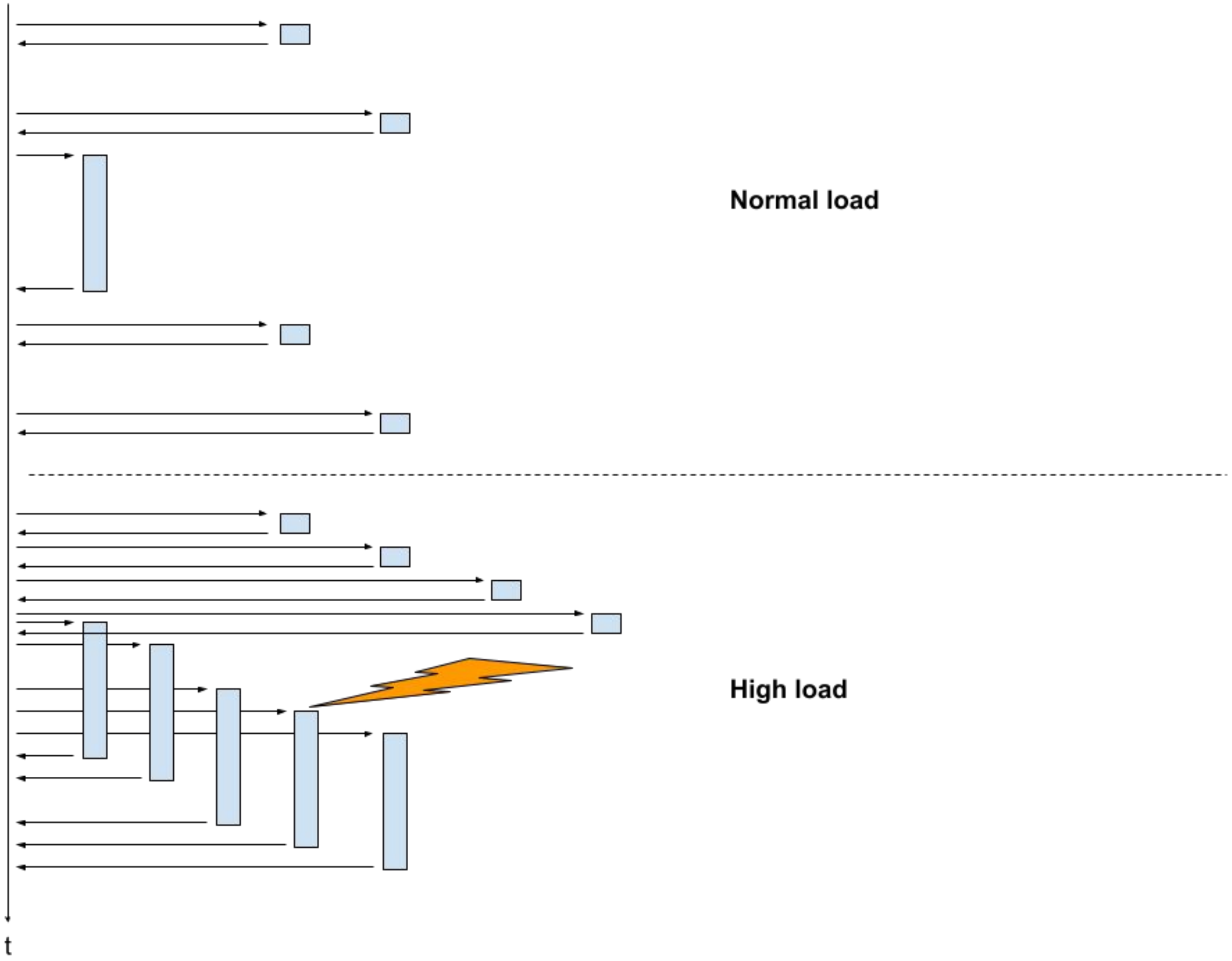
- Idea:
  - 1 full page = 1 cache item
  - Endless TTL
  - Purge cache for page when data changes

- Pitfall:
  - Render menu / links dynamically from page structure
  - →purge entire cache on change of every page

ORM: Cache single entities

- Idea:
  - Build cache into object relational mapping
  - Cache each entity by its ID
- Pitfall:
  - Fetching lists of entities: Needs full fetch + replacement of entities by ID
  - Partially fetched (complex) entities pollute the cache with incomplete data

Under load: Cache stampede



Dynamic route cache

- Idea:
  - Routes are needed in every request
  - They are compiled from the node tree
  - Node tree changes only on incoming replication
  - →Compile & cache routes on incoming replication only (**local server!**)
- Pitfall:
  - With few instances: likelihood an instance receives replication is high
  - With many instances: instances end up with outdated route cache

Sitemap rendering

- Idea:
  - Sitemaps are crawled only rarely
  - Generating them takes much time
  - →Run generation nightly on every server and store static files
- Pitfall:
  - Few deployments + load peeks
  - Newly created instances have outdated sitemaps

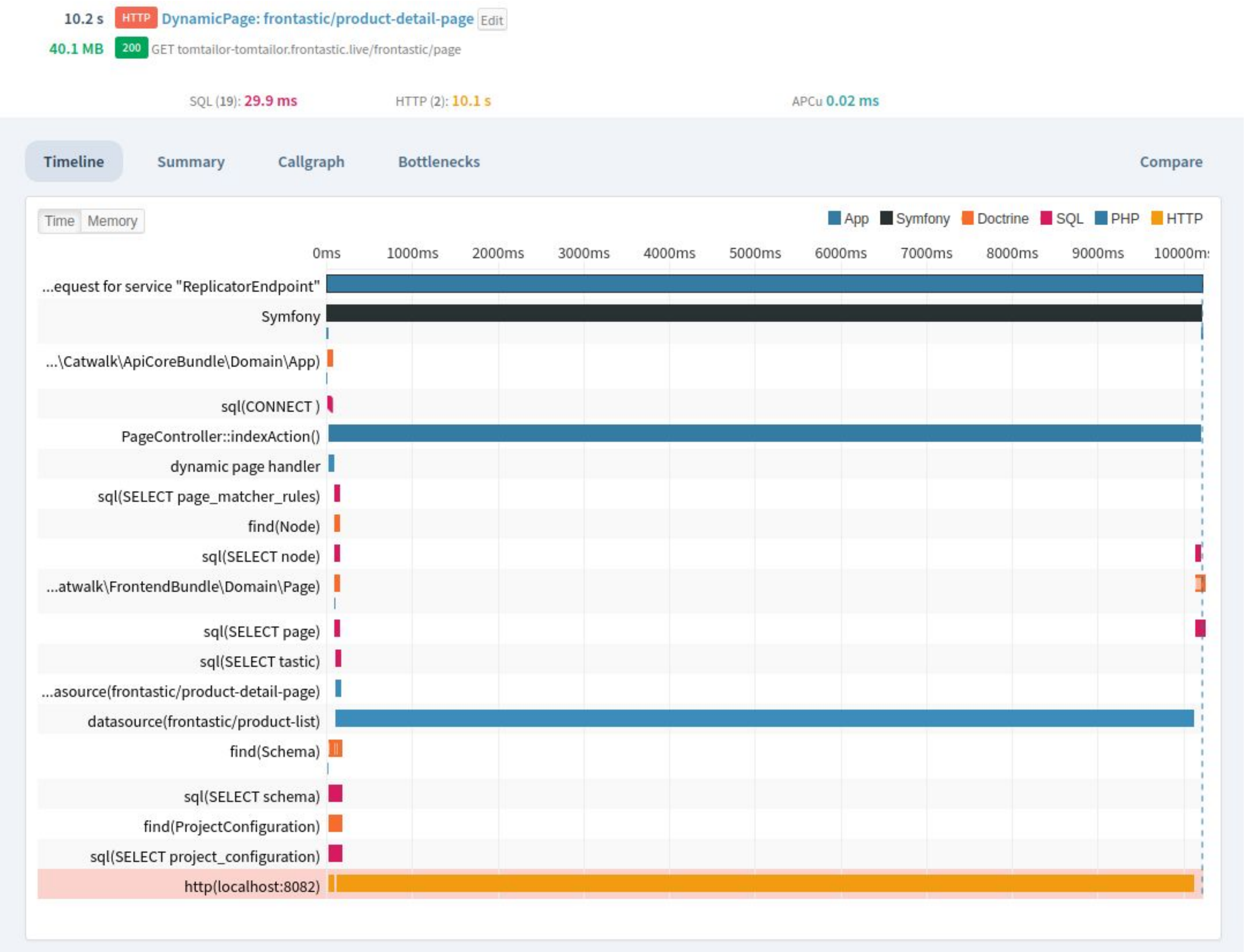
Lessons to be learned

… or how to prevent caching in the first place

What problem do you actually want to solve?

- Slow code execution
- Frequent request scaling
- Large resource consumption
- …

Analyze the cause of the problem!



10.2 s HTTP DynamicPage: frontastic/product-detail-page Edit

40.1 MB 200 GET tomtailor-tomtailor.frontastic.live/frontastic/page

SQL (19): 29.9 ms HTTP (2): 10.1 s APCu 0.02 ms

Timeline
Summary
Callgraph
Bottlenecks
Compare
Time
Memory
App
Symfony
Doctrine
SQL
PHP
HTTP
0ms
1000ms
2000ms
3000ms
4000ms
5000ms
6000ms
7000ms
8000ms
9000ms
10000m
...equest for service "ReplicatorEndpoint"
Symfony
...\Catwalk\ApiCoreBundle\Domain\App)
sql(CONNECT)
PageController::indexAction()
dynamic page handler
sql(SELECT page_matcher_rules)
find(Node)
sql(SELECT node)
...atwalk\FrontendBundle\Domain\Page)
sql(SELECT page)
sql(SELECT tastic)
...asource(frontastic/product-detail-page)
datasource(frontastic/product-list)
find(Schema)
sql(SELECT schema)
find(ProjectConfiguration)
sql(SELECT project_configuration)
http(localhost:8082)

Try to solve the issue without caching

- Database indexes
- Algorithmic improvements
- Make code asynchronous

If you cannot come around caching …

Design your cache thoughtfully!

- Choose a single layer for caching
- Try to keep the cache outside of the system
- Gather currentness expectations
- Calculate cache dimensions & buy enough RAM
- Create a proper system setup for your cache
- Take measurements to prevent typical issues like cache stampede

Conclusion

There are 3 essential challenges in computer science:

- Caching
- Off-by-one bugs

Questions? Answers!

Get the slides:
https://schlitt.info