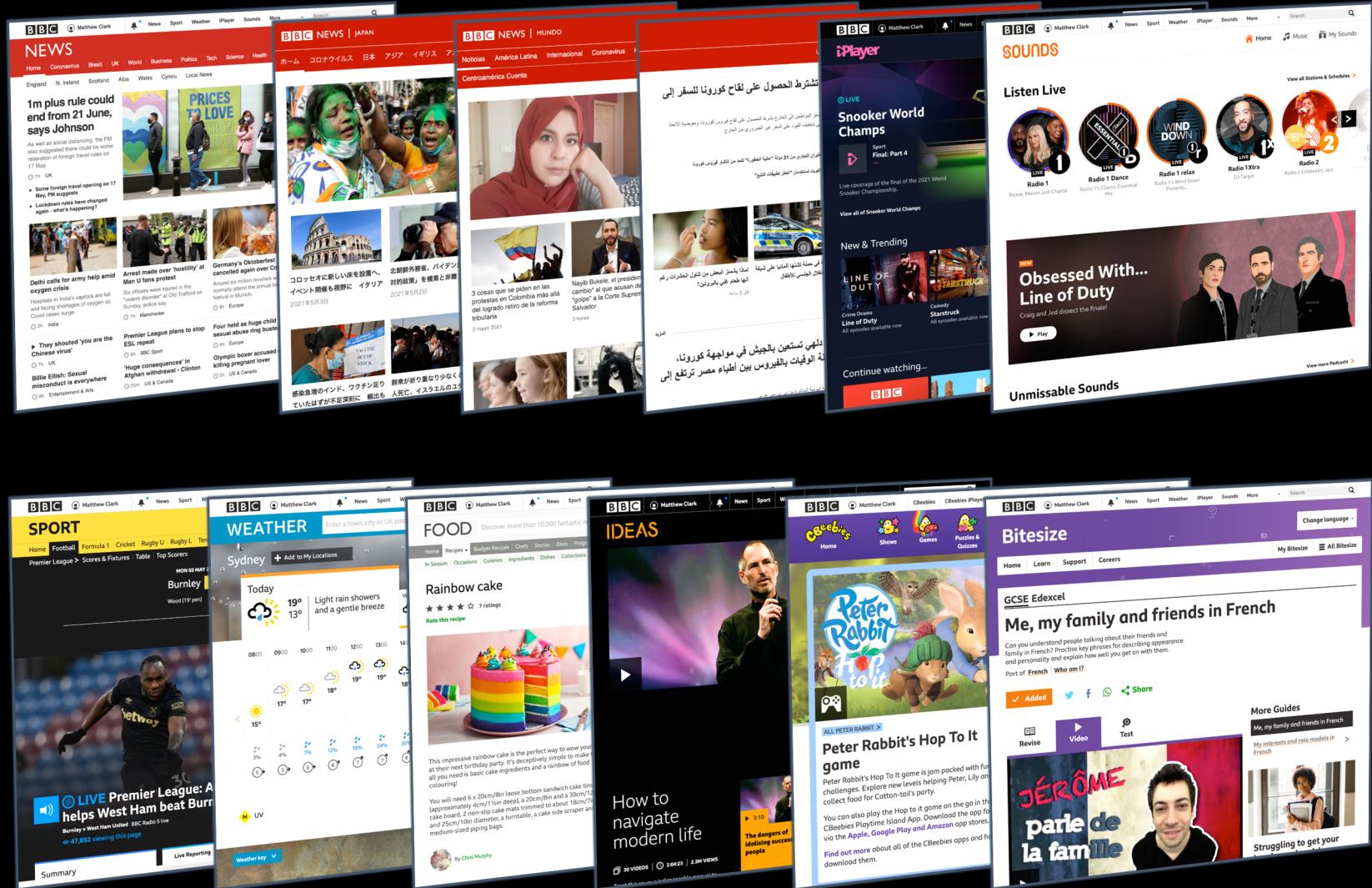


Architecting for scale with the cloud and serverless

A case study of BBC Online

Matthew Clark
@matthew1000



Matthew Clark
@matthew1000

Architecting for scale with the cloud and serverless

QCon+ May 2021

Our requirements



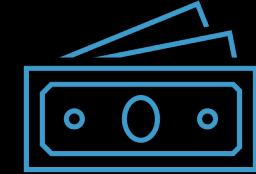
High traffic

Global and
highly variable



**Broad range
of content**

Created by
multiple teams

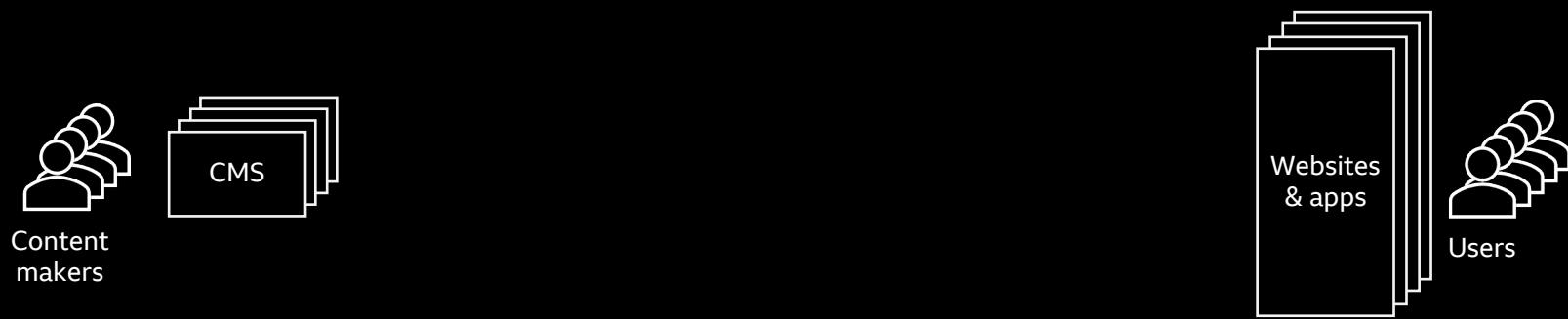


Good value

Quick to build,
affordable to run

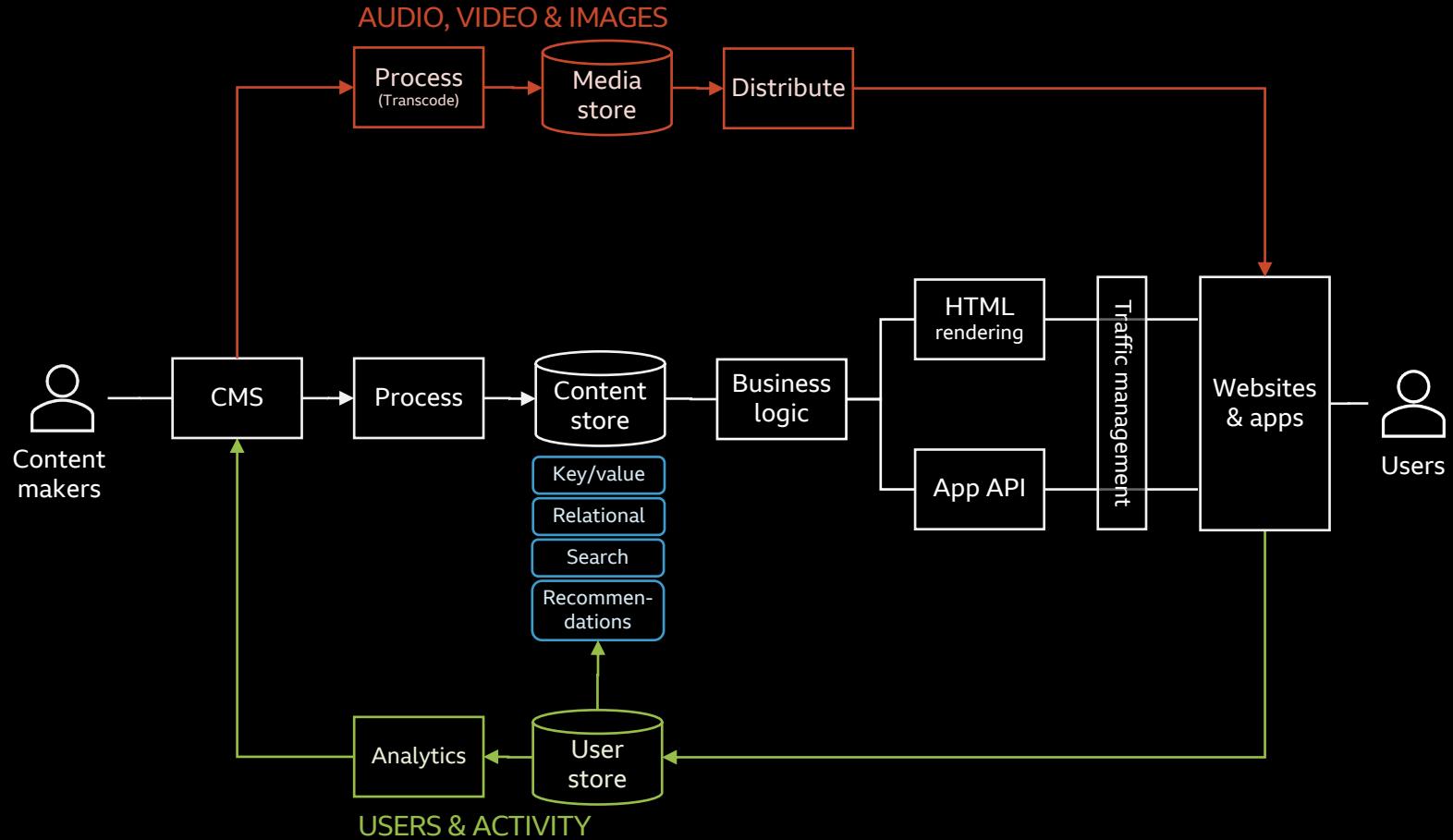
Matthew Clark
@matthew1000

Architecture



Matthew Clark
@matthew1000

Architecture



Matthew Clark
@matthew1000

Caching

Four reasons to cache:



Scale

Handle more requests



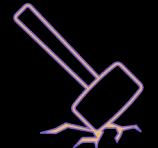
Speed

Faster responses



Cost

Usually cheaper overall

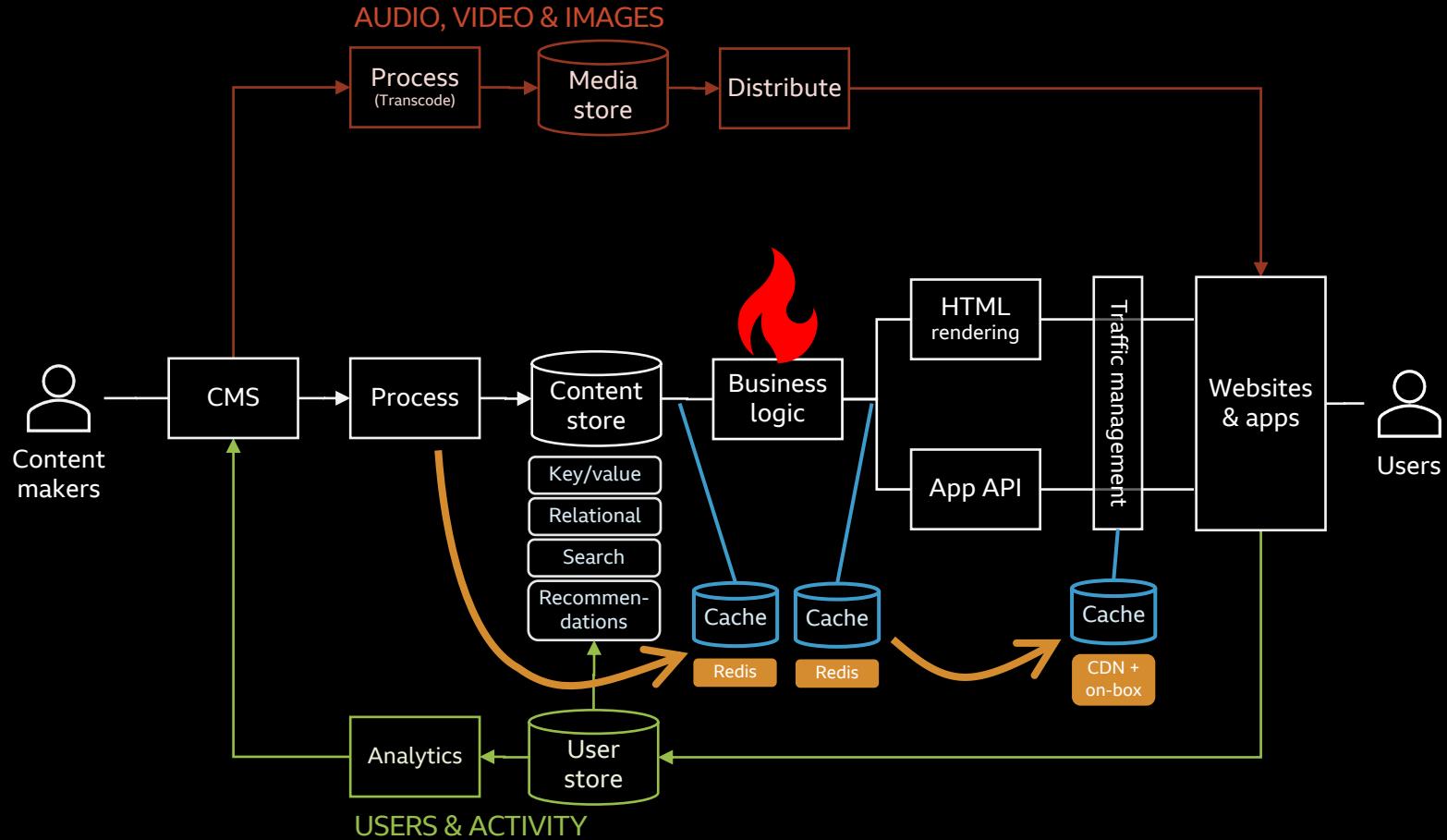


Resilience

Serve stale on error

Matthew Clark
@matthew1000

Architecture



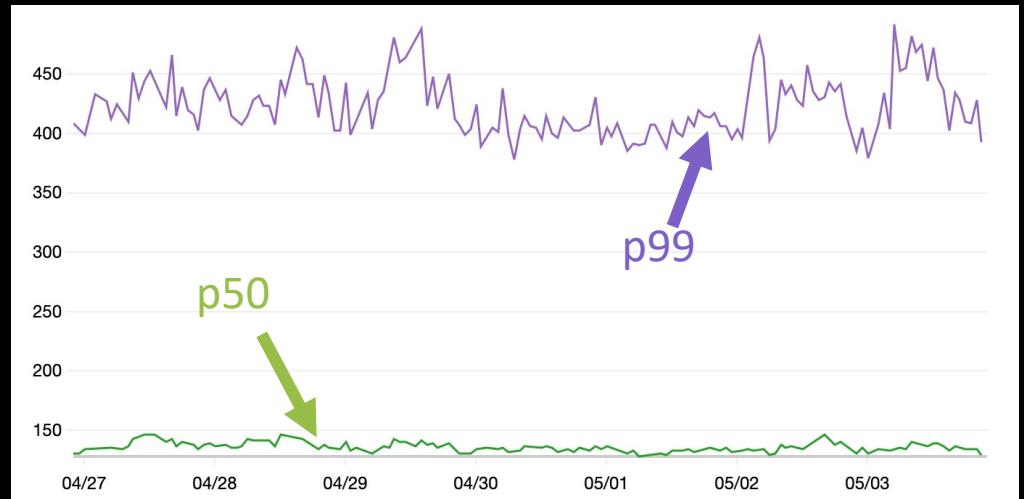
Matthew Clark
@matthew1000

Serverless (compute & more)

↑
P100
multiple
seconds

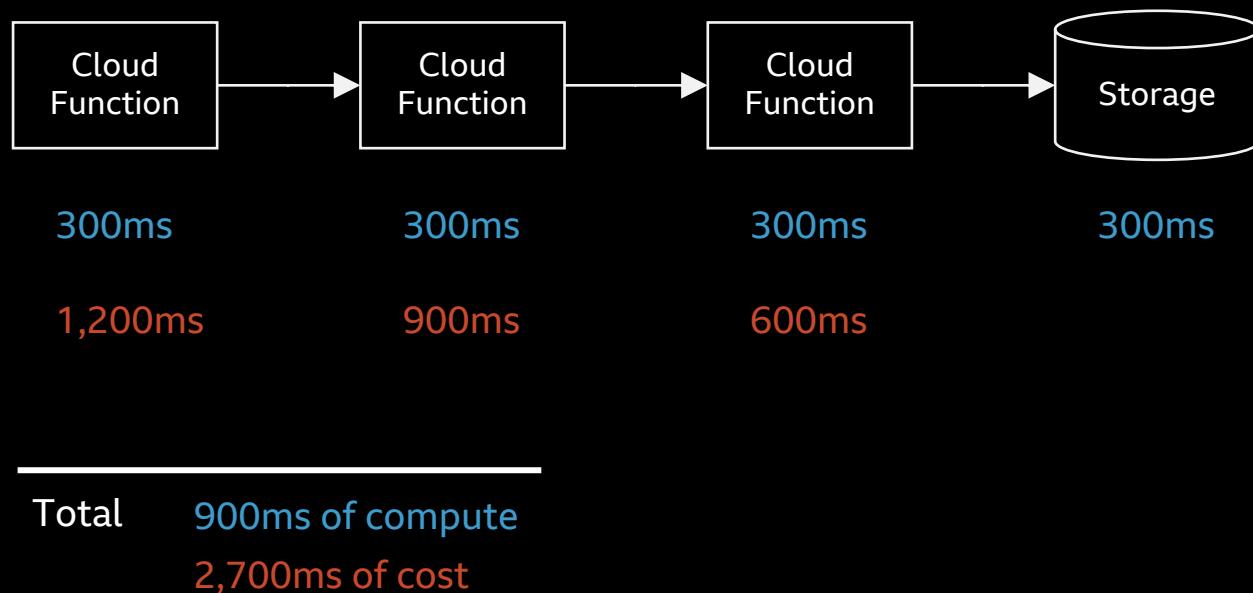
The three arguments against serverless:

1. Slow
2. Restrictive
3. Expensive



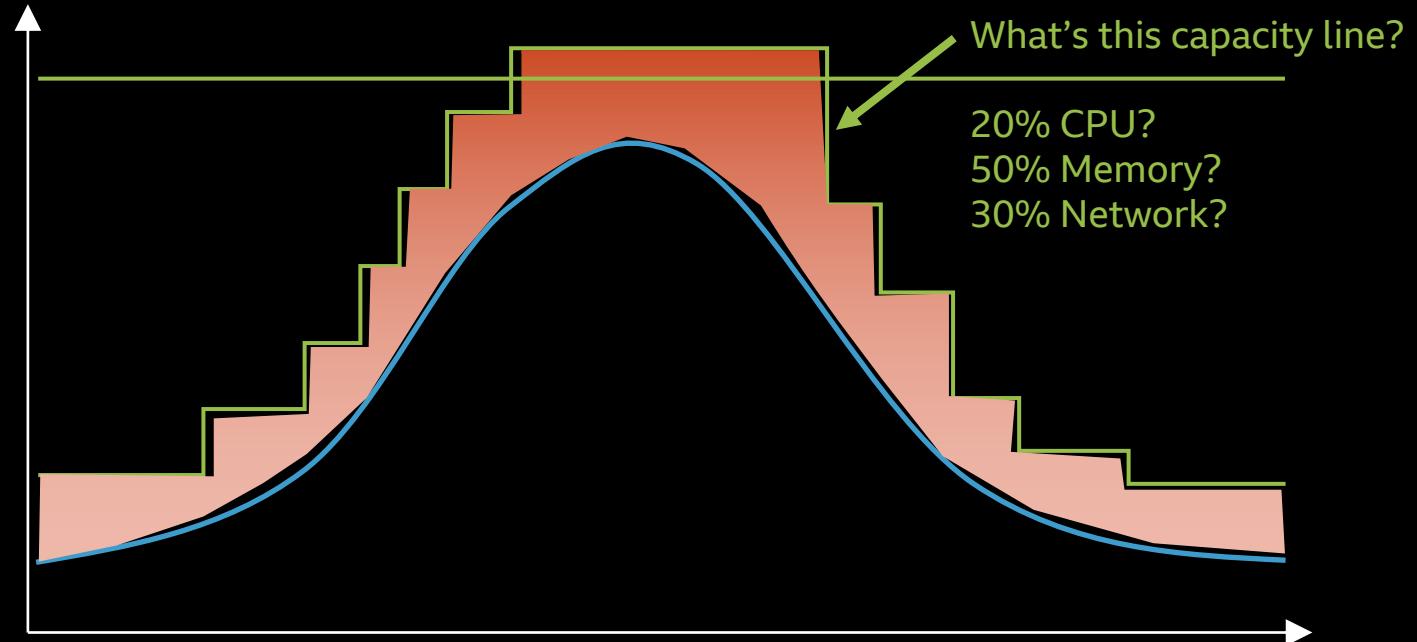
Matthew Clark
@matthew1000

Idle function problem (aka function chaining)



Matthew Clark
@matthew1000

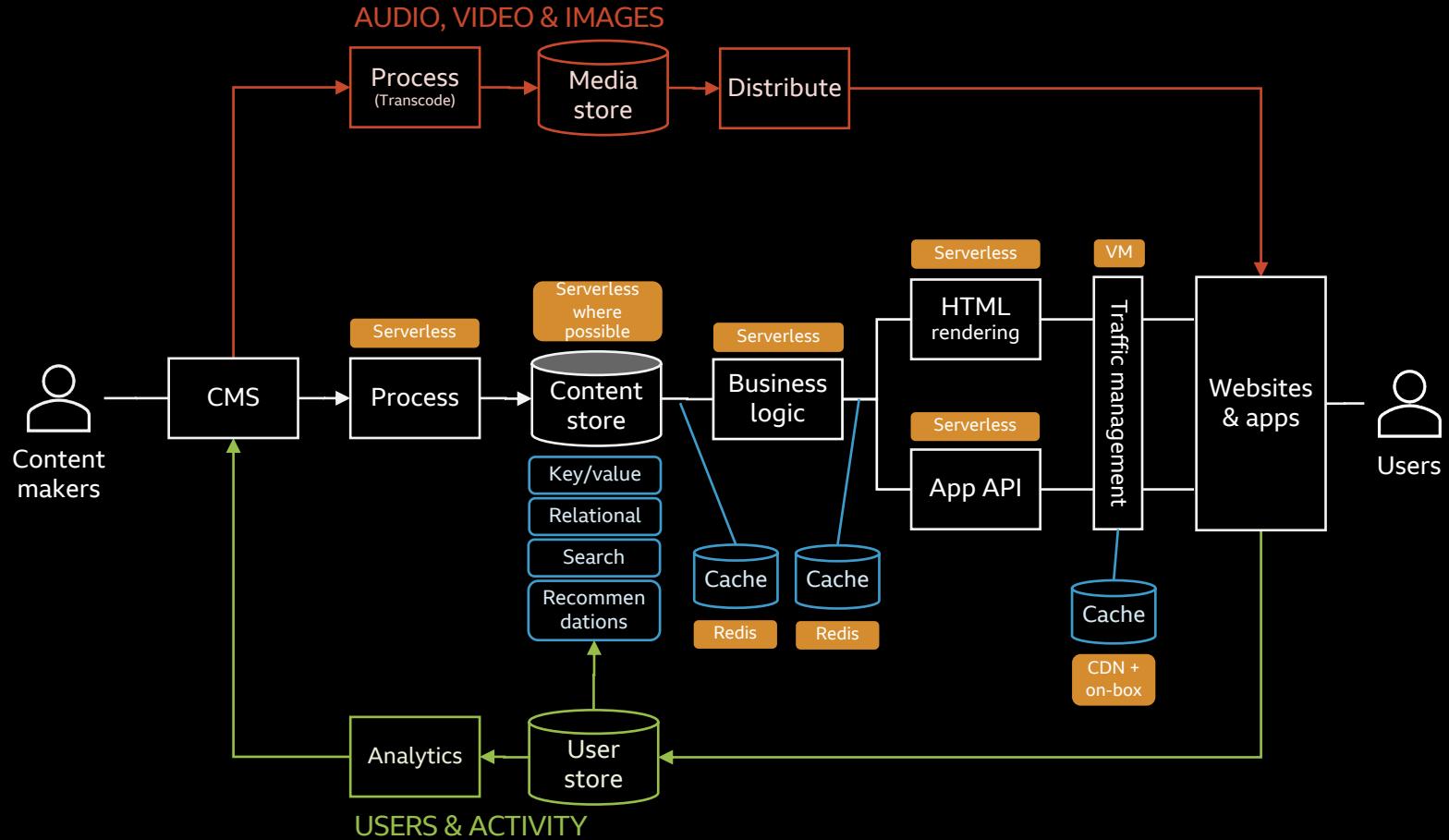
Autoscaling is, in effect, permanent overprovisioning



Matthew Clark
@matthew1000

Serverless compute may cost more, but you need less

Architecture

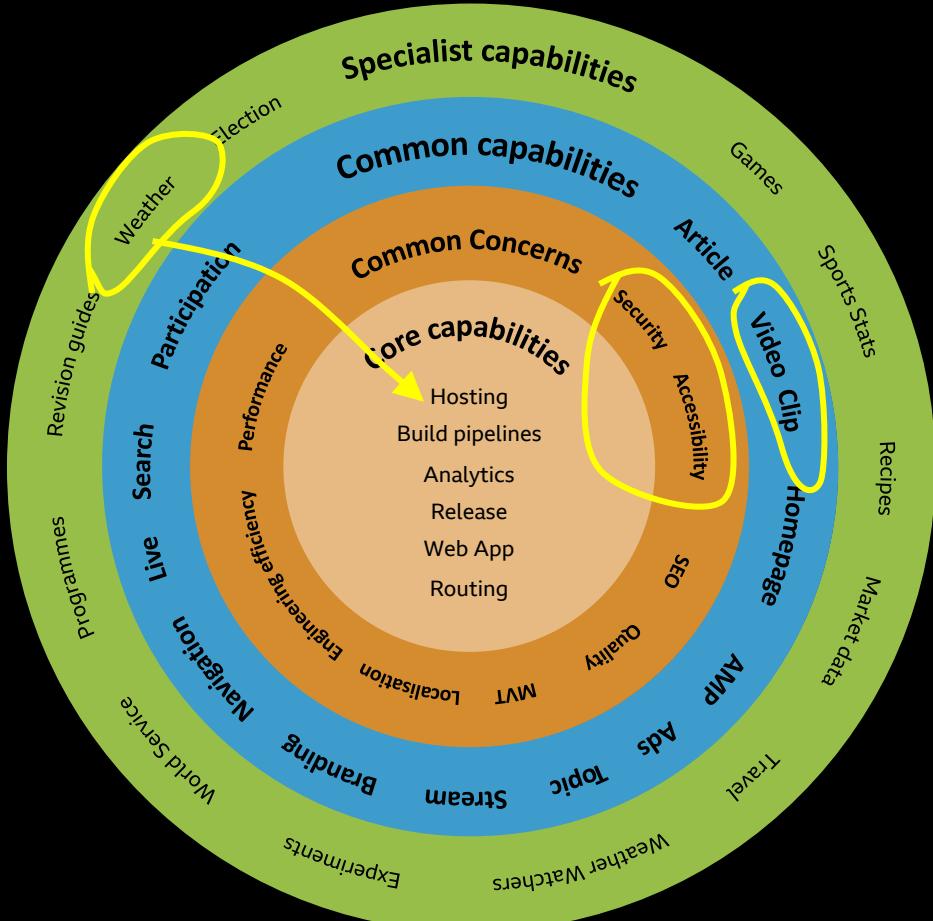


Matthew Clark
@matthew1000

Have clear ownership

Matthew Clark
@matthew1000

Have clear ownership



Make a list of everything that needs to happen.

Then organise around it.

Ensure each item is clearly owned end-to-end by a team.

Matthew Clark
@matthew1000

**Make the common things easy
and the specialist things possible**

Matthew Clark
@matthew1000

Make the common things easy and the specialist things possible

80% of services take
the golden path

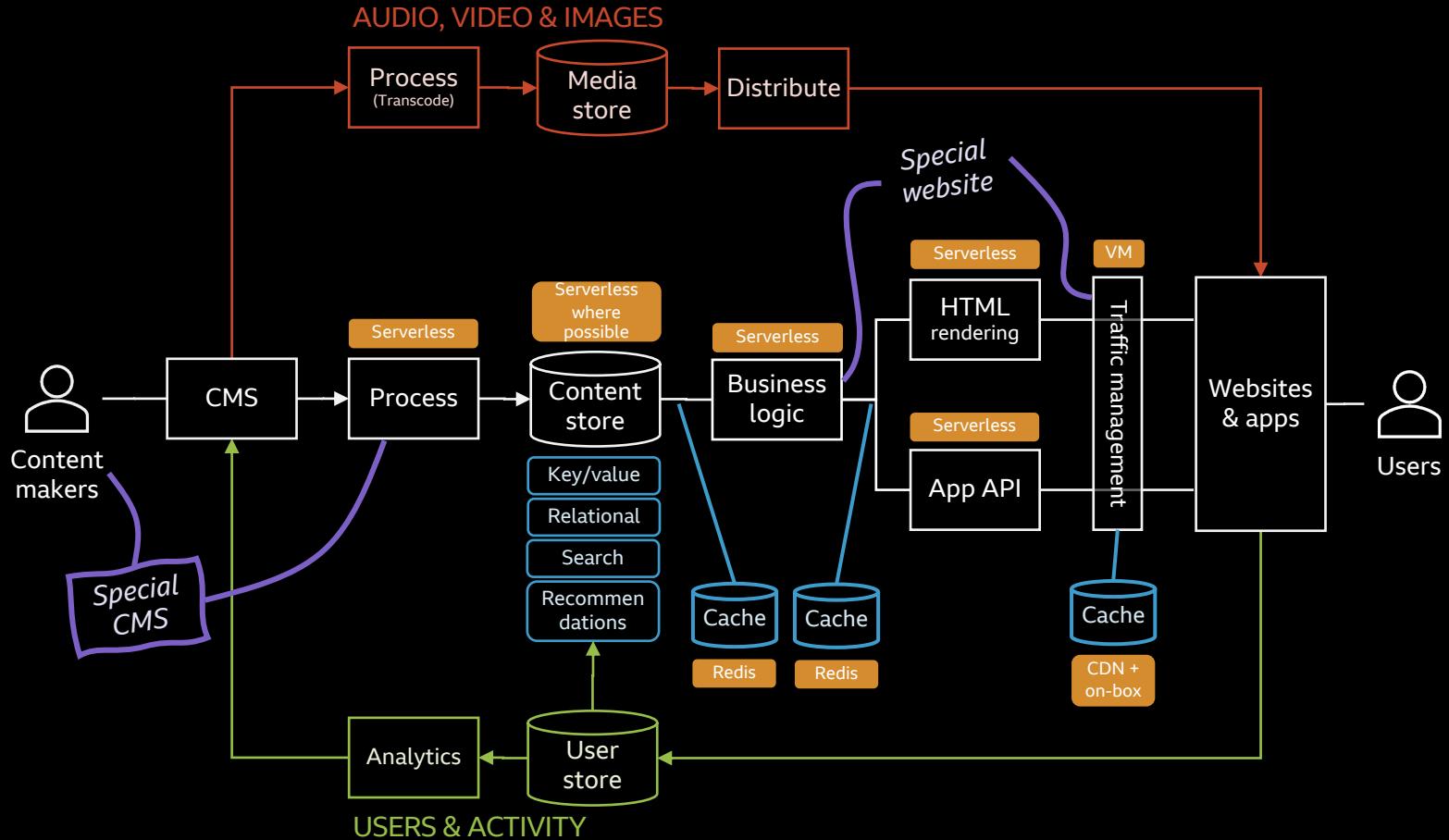
- Consistent technology choices
- Shared tooling
- Maximise reuse

20% of services have
specialist requirements

- Different technologies
- Different tooling
- Limited reuse

Matthew Clark
@matthew1000

Architecture



Matthew Clark
@matthew1000

Summary

Architecture

```

graph LR
    CMS[Content makers] --> Process[Process]
    Process --> MediaStore[Media store]
    MediaStore -- Distribute --> Websites[Websites & apps]
    CMS --> ContentStore[Content store]
    ContentStore --> BusinessLogic[Business logic]
    BusinessLogic --> AppAPI[App API]
    AppAPI --> Websites
    CMS --> Analytics[Analytics]
    Analytics --> UserStore[User store]
    UserStore --> Recommendation[Recommendations]
    Recommendation --> Cache[Cache]
    Cache --> Redis[Redis]
    Cache --> Memcached[Memcached]
    Cache --> CDN[CDN + Origins]
    Redis --> CDN
    CDN --> Websites
    
```

Autoscaling is, in effect, permanent overprovisioning

Serverless compute may cost more, but you need less

Organise around the problems

Make a list of everything that needs to happen.
Then organise around it.
Ensure each item is clearly owned end-to-end by a team.

Thanks for watching

Thoughts most welcome, on Twitter, LinkedIn, or matthew.clark@bbc.co.uk
We're hiring too (in the UK) ☺

Matthew Clark
@matthew1000