

# 数千万PVをさばくためのインフラ構成について

2022/06/08 TomoakiTANAKA

# 自己紹介

# 田中智章

- 富岡市出身
- SE としてキャリアを始め、現在は事業会社勤務
- 主に Web開発 を担当、時々アプリ開発 (Rails / Flutter / Unity ← New)
- 好きなお酒はウィスキー全般、あと LEGO が好き



# 宣伝

## 会社/プロダクト紹介

# 株式会社エバーセンス

## Vision

- 家族を幸せにすることで、笑顔溢れる社会をつくる。

## Products

- 妊婦さんへの情報提供をする  
「ninaru」
- 子育てに必要な情報や機能満載の  
育児アプリ 「ninaru baby」



勉強会資料●数五万PVをさぼくたとえのとオマフリ社のSVとして、2022年4月にできたばかりの会社

- 児童向けの知育アプリの開発等を行っています

## Products

- タッチであそぼ！あかまるどれかな？
- iOS / Android 版があるので小さなお子様やお孫さんがいらっしゃる方は是非



by Tomoaki TANAKA

こどもたちが自由に

人生を歩む社会を

つくる



今、こどもたちには自分の人生を拓くための選択肢がたくさん用意されています。

その選択肢にこどもたちが自ら気づくこと。  
そして、自分の力で人生を歩む彼らを周りの大人が応援する社会にしたい。

DanRanはデジタルとコンテンツを通じ、あらゆる大人を巻き込みながら、こどもたちが可能性を広げる機会をつくりだしていきます。

## アイスブレイク

- Q1. PVってなんのことでしょうか？（なんの略でしょうか）
- Q2. Webメディアってなんでしょうか？
- Q3. 月間PV 4,500万 (@todo: 今いくつだ？) ってすごいのでしょうか？

是非スレッドに回答をお願いします

# アイスブレイク シンキングタイム

- 勉強会資料 数千万PVをさばくためのインフラ構成について
- A1. PVはPageView（ページビュー）の略で、Webページの閲覧回数のこと
  - A2. www（WorldWideWeb）上で、記事をまとめた読めるWebサイト（Webサービス）のこと
    - 例) Qiita, zenn
    - 例) マイナビニュース, ロケットニュース24
  - A3. 月間PV 4,500万（@todo: 今いくつだ？）ってすごいのでしょうか？
    - 月間平均で1,000pv/分なので、結構すごいと思います
    - GoogleAnalyticsのリアルタイムユーザーでいうと、同時接続3,000人くらいです（でした）
    - もちろん上を見たらきりがないですが...
      - 例) Yahoo!ニュース 225億PV/月
      - 例) zozoタウン 5億PV/月

出典：

- 1つの記事で世の中が大きく変わる——1日の記事数約6000本、月間225億PVを数える

# 本日の勉強会について

## 本日の勉強会について

中規模「Webメディア」を効率的に運営する方法に関して、考え方や知見をお話します。

あくまで田中個人の知見や経験によるものですので、取り扱いにはご注意ください。

掲載内容は私自身の見解であり、必ずしも所属する企業や組織の立場、戦略、意見を代表するものではありません。

- 数百万～数千万PVのWebメディアを、できるだけ安いインフラ構成でさばくための考え方
- 下記のシステム構成を前提にした話をします
  - サーバーサイドレンダリングを前提としたシステムにおける
  - AWSを前提としてインフラ構成（多分GCPやAzuleでも通用する考え方）

## 話さないこと

- 数百万～数千万PVのECサイト等、WebメディアではないWebサービスの大規模トラフィックをさばく技術
  - 例) EC, 不動産サイト, etc
- オンプレを前提としてインフラ環境における構成
- フロントエンドレンダリング全般
- 具体的なプライシングやインスタンス構成

## この話を聞いて得られそうなこと

- Webサービスの運用を見据えたインフラ構成やテクニックの理解
- テレビCMやインフルエンサーによる急なアクセス増加に対する心の安寧（個人差あり）

# 目次

# 目次

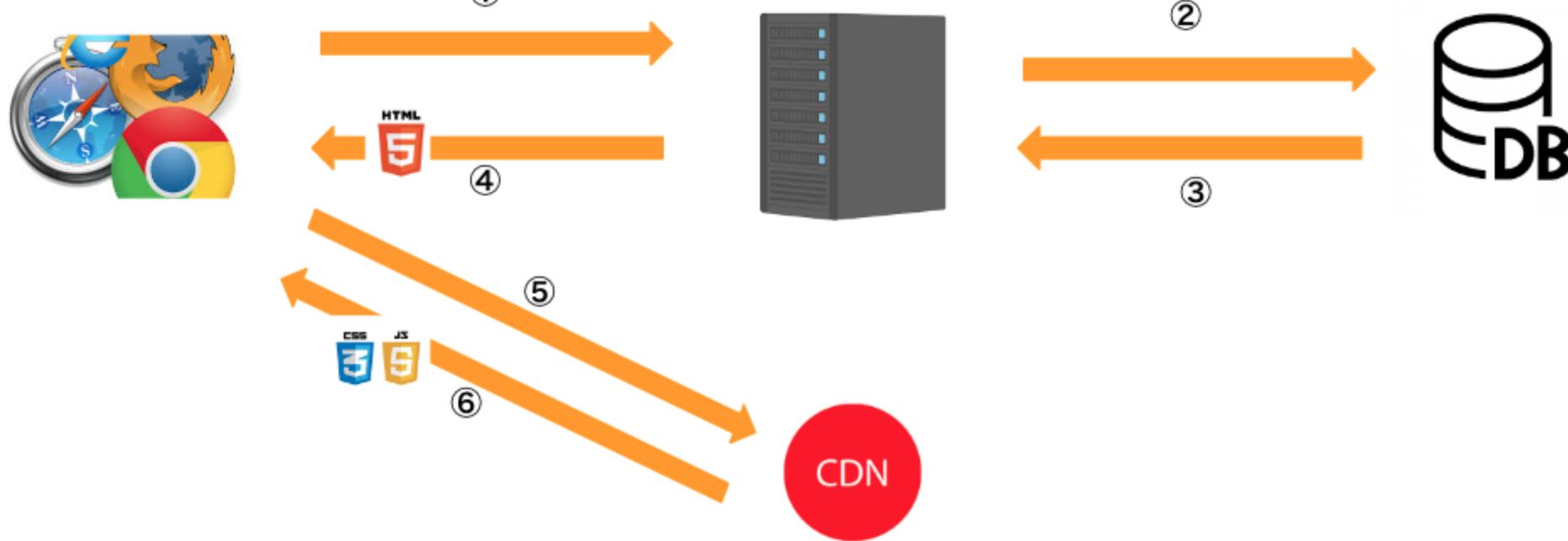
1. Webメディアの特徴と構成技術の分解
2. キャッシュを使ってアクセスをさばく（事例紹介）
3. まとめ

\* 組織の意見でなく、個人の意見です

# Webメディアの特徴

## Webメディアの特徴

- 誰が見てもほぼ同じ構成
  - 記事はだれが見てもほぼ同じ
  - 広告やレコメンドはユーザーによって異なる
- 技術的に分解すると...
  - 記事はだれが見てもほぼ同じ (HTML / CSS 部分)
  - 広告やレコメンドはユーザーによって異なる (JavaScript部分)
  - 読者にとっては、HTTPのGET中心の世界 (データを取得するのが主)



\* サーバーサイドレンダリングを前提にしています

- ①: ブラウザからWebサーバーへリクエストがはいる
- ②: サーバーは必要に応じて、DBへアクセス。データを取り出す
- ③: 略
- ④: サーバーは、テンプレートエンジン等を使ってHTMLを作成、ブラウザに返す
- ⑤: ブラウザはHTMLを解釈、必要に応じてcssやJavaScriptをCDN問い合わせ

名前	読込速度	読込速度(MB/s)
メモリ	10GB/s	10000MB/s
有線ネットワーク	1GB/s	1000MB/s
SSD(シーケンシャルアクセス)	100MB/s	100MB/s
HDD(シーケンシャルアクセス)	100MB/s	100MB/s
SSD(ランダムアクセス)	10MB/s	10MB/s
HDD(ランダムアクセス)	1MB/s	1MB/s

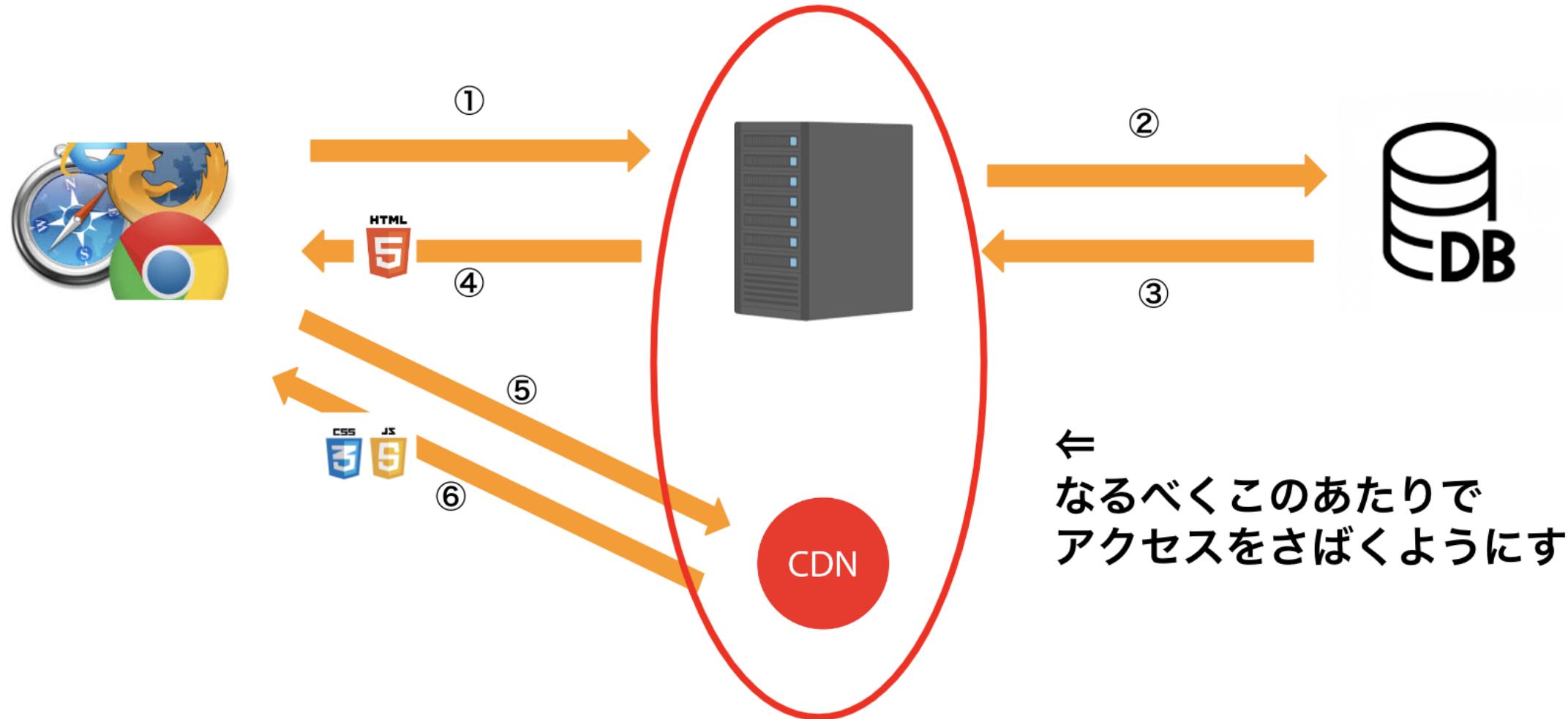
## ボトルネック解消をどのように考えるか？

- できる限りWebサーバー側でリクエストをさばく
- できる限りWebサーバーも、メモリやHDDを減らしたい（安くなるから）

### ボトルネック

- お値段：DB（例：RDS） > Webサーバー（例：EC2）
- お値段：メモリ > HDD
- 転送速度：メモリ > HDD > ネットワーク

# ボトルネック解消をどのように考えるか？



## キャッシュを使ってアクセスをさばく（事例紹介）

# キャッシュとは？

キャッシュは、CPUのバスやネットワークなど様々な情報伝達経路において、ある領域から他の領域へ情報を転送する際、その転送遅延を極力隠蔽し転送効率を向上するために考案された記憶階層の実現手段である。実装するシステムに応じてハードウェア・ソフトウェア双方の形態がある。

↓↓↓

- よく使うデータやファイルを、アクセスしやすいところにおいておく
  - 例) アクセスしやすい = 高速に取り出せる場所（メモリやローカルファイル）
- 逆にあまり使わないファイルは、必要なときに取り出す
  - 例) 必要なときにネットワーク経由で取り寄せる

出典：

- [キャッシュ](#)

## Webメディアによるキャッシュ戦略

- よく使うファイル
- クエリなしのURLに対するHTMLファイル
- CSSやJavaScriptなど、共通でアクセスするファイル
- あまり使わないファイル
  - クエリパラメータ付きのURL（検索やページネーション）
  - GET以外のリクエスト

## Webメディアによるキャッシュ戦略

- クエリなしのURLに対するHTMLファイル
- => WebサーバーでHTMLファイルを事前作成（ファイルキャッシュ）
- CSSやJavaScriptなど、共通でアクセスするファイル
  - => CDNで高速アクセスできるようにしておく（S3 + CloudFront）

## ファイルキャッシュの運用事例

# 運用例 / 設定例（キャッシュの作成、削除）

## コーポレートサイトのようなアクセスが少ないサイトの場合

- キャッシュの作成
  - Webサーバーのデフォルト機能を使う（アクセスがあったらキャッシュを作成する）
- キャッシュの削除
  - 記事公開 / サイト全体の更新時（css変更など）
  - まとめてファイルを消す

### まとめ

- 例えばWordPressには、まとめてファイルキャッシュを削除してくれるようなプラグインがあるのでそれを利用する

# 運用例 / 設定例（キャッシュの作成、削除）

勉強会資料 数千万PVをさばくためのインフラ構成について

## Webメディアのようにアクセスが多いサイトの場合

- キャッシュの作成
  - Webサーバーのデフォルト機能を使う（アクセスがあったらキャッシュを作成する）
- キャッシュの削除
  - 記事公開
    - => 記事のURL、トップページ、記事のカテゴリやタグページなど、関連ページを個別に削除
  - サイト全体の更新時
    - => URL一覧を取得して、数十～数百ページ単位でキャッシュを作成

## まとめ

- 常時アクセスが多いサイトだと、一度にファイル削除はキャッシュが薄くなるので危険
- キャッシュ生成や削除の部分は、基本手作り

# 運用例 / 設定期 (キャッシュさせる / させない)

勉強会資料 数千万PVをさばくためのインフラ構成について

- とはいっても全部のファイルをキャッシュさせてしまうと運用上困る
  - 例) 編集者が記事を更新したのに最新版が反映されない（確認できない）
  - 例) sitemapなど、リアルタイムでGoogleに伝えたいものが、古いままだと困る

@nginx/configure/cache

```
set $do_not_cache 0;

# WordPressの管理画面にログインすると付与されるcookieを持っていたらキャッシュなしにする
if ($http_cookie ~* "wordpress_(?!test_cookie)|wp-postpass_" ) {
    set $do_not_cache 1;
}

# GET以外はキャッシュなしにする
if ($request_method != GET) {
    set $do_not_cache 1;
}

# sitemapなど特定のURLはキャッシュなしにする
if ($request_uri ~* "(sitemap.xml") {
    set $do_not_cache 1;
}
```

# CDNの話

基本的に、assetファイル（画像やcss等）をファイルサーバーに配置し、CDN登録するだけとはいえ、トラブルも…

- 例) 画像ファイル更新したのに、更新されない
- 原因
  - S3のファイルは変更されているが、CDNのキャッシュが残るため
- 対策
  - フィンガープリントを使ってデプロイのたびにファイル名を書き換える
  - CDNのキャッシュを削除する

フィンガープリントの例

```
app.css  
↓↓↓  
app-98f2b4256f620be0496ff18f157f863a.css
```

# まとめ

## まとめの前に

Q.

- そもそもGET中心でいいなら、HTMLファイルごとCDN配信すればよいのでは？

A.

- まさしくそのとおり
- 最近のミニサイトであれば、ローカルでHTMLを作成して、s3にアップロード => CDN配信の形もおすすめです

## まとめ

- Webメディアの特徴を説明しました
  - GET中心の世界
- Webメディアにおけるアクセスのさばき方を説明しました
  - 基本はファイルキャッシュ
- ファイルキャッシュやCDNの具体的な運用について説明しました

ご清聴ありがとうございました

## 質疑応答

おわり