CONTENTS

1

はじめに

1.1 本書の対象と目的

本書はHTTPを介した配信とその最適化について解説する入門書です。配信に従事した経験のないアプリケーション・インフラエンジニアでもわかるように、初歩から説明していきます。

配信と言われても、言葉の意味も広く、何が何だか想像もつかないという方もいるかもしれません。本書における配信とは、Webにおいて（主にHTTPで）コンテンツ[*1]をサーバーからクライアントに届けることを指します。本書では配信をより高速・安定・安全にすることを配信最適化と定義します。配信を実現するさまざまな手段を配信技術、配信に関連するリバースプロキシ（Proxy）[*2]やCDN[*3]などを用いて構築した一連のシステムを配信システムとします。

配信の安定性や速度はユーザー体験に直結します。Webサイトや Webサービスを提供するうえで、避けては通れない領域です。

配信技術はサーバー・クライアント間にわたります。単純にどこか1つの知識があればよくできるものではありません。全体像を把握し、適切にアプケーションを設定し、インフラ構成を組み、キャッシュを行い、場合によってはCDNなどの外部サービスを使うなど対応は多岐にわたります。

とはいっても、現段階では何のことやらでしょう。ごく単純な構成で配信を意識して改善すると、どのように役に立つかを考えてみましょう。素朴なWebサイトやブログを立ち上げる際にはレンタルサーバーなどにサイトを設置することが多いです。

サーバーはインターネットにつながっており、スマーフォンやPCなどの**クライアント**はインターネットを経由してWebサイトを閲覧しに来ます。

＊1　本書では、サーバーのローカルファイルをコンテンツリソース、サーバーのコード上で生成されたいわゆる動的なWebページなどとコンテンツリソースを合わせたものをコンテンツと定義します。なおリソースと単体で示したときは、CPUやストレージなどの資源を示します。

＊2　プロキシサーバーの一種。本書では基本的にProxyと記載。Apache HTTP ServerなどWebサーバーや、アプリケーションサーバーの前段（クライアント側）に立て、クライアントからのリクエストをまとめてWebサーバーやアプリケーションサーバーに投げます。クライアントからのリクエストを整理したり、アプリケーションサーバーの出力内容をキャッシュしたりします。詳細は4章で解説します。

＊3　Content Delivery Networkの略称。Webのコンテンツ配信をより安定化、より高速化するためのサービス。代表的なサービスにAkamaiやFastlyなどがある。

3

HTTPヘッダ・設定とコンテンツの見直し

配信に問題を見出したとき、おそらく最初に思い浮かぶ対策はCDN導入やサーバー増強でしょう。

ところが、少なくないサイトで、これらの対策にほとんど効果がないことがあります。このとき、問題はCDN導入やサーバー強化以前にあります。

- 標準仕様の誤解から不用意な設定を行っていることでうまくローカルキャッシュが使えていない
- 小さく表示するためのサムネイルなのに明らかに過大な画像が設定されている*1
- ……

こんな事例を数多く見てきました。CDN導入や大規模な構成変更に手を付ける前に、改善すべきポイントは数多くあります。こういった改善を一つ一つ積み重ねることがサービスの快適性、安定性やコスト削減につながります。

本章では、こういった**大幅な構成変更を行わずにできる効率化**を解説していきます。

配信を行う環境はさまざまですが、総転送量の削減が利益に大きく影響する、という点はおおよそ共通しています。たとえばAWSをはじめとするクラウドサービスでは、転送量ベースの課金を採用しているものがあります*2。無計画に配信していると、ここが馬鹿にならない金額になります。

転送量（総転送量）は次のように求められます。

```
総転送量 = リクエスト数 x 平均コンテンツサイズ
```

本章では、次の2点を軸に、総転送量の削減を目指します。

- 標準仕様にのっとったHTTPヘッダの設定を行い、ローカルキャッシュをう

*1　筆者はサムネイルで8K（7680×4320）ものサイズの画像と遭遇したことがあります。
*2　サービスによってはリクエスト数課金などもある。

4

キャッシュによる負荷対策

前章で、クライアントに保存されるローカルキャッシュを有効に使えるようになりました。これで負荷対策は完璧でしょうか？

実際に起こりうるケースで考えてみましょう。突然SNSでサービスが話題になってユーザーが増えたらどうでしょう？ 2章で触れた通り、ローカルキャッシュはそのクライアントのみで再利用されます。つまり、すでにサイトを見ているクライアントには有効です。

ところがユーザーが増えるということは、新規の、ローカルキャッシュを持っていないクライアントからのリクエストが増えるということです。当然、ユーザーが増えれば増えるほどオリジンへのリクエストが増え、直撃する負荷は上昇します。ここまで主に解説してきたクライアント側のキャッシュだけでは、ユーザー数の単純な増加にはどうやら対応できなさそうです。

ここで役に立つのがProxyやCDNで行う、経路上のキャッシュです。

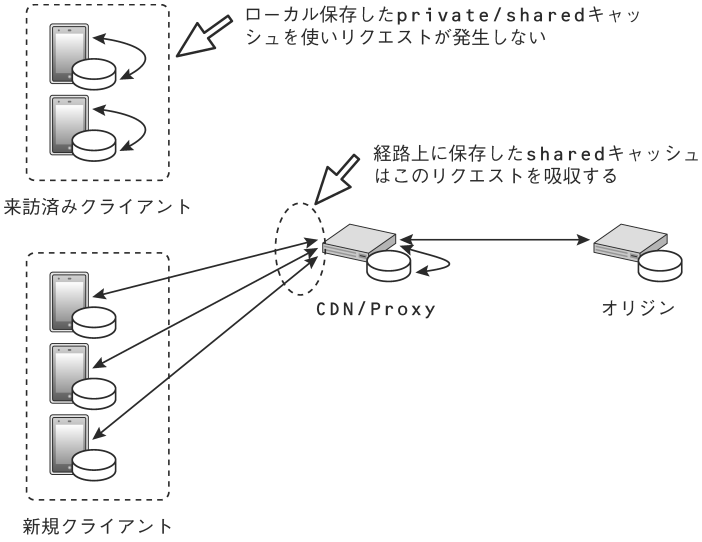

図4.1 ローカルと経路上のキャッシュが効くところの違い

図のように新規クライアントからのリクエストもProxy/CDNが経路上のキャッシュを持っていれば、そこからレスポンスします。持っていなくても、オリジン

6

CDNを活用する

全世界のユーザーに向けて配信を行いたい、トラフィックが非常に大きい配信を行いたいという要件は、Webサービスを運営していれば当然のように出てきます。サイトが成長して、トラフィックの伸びに対して、既存構成だと耐えきれないので何とかしたいということもあるでしょう。

その際に便利に使えるのが、ここまで何度も名前を出してきたCDN（Content Delivery Network）です。

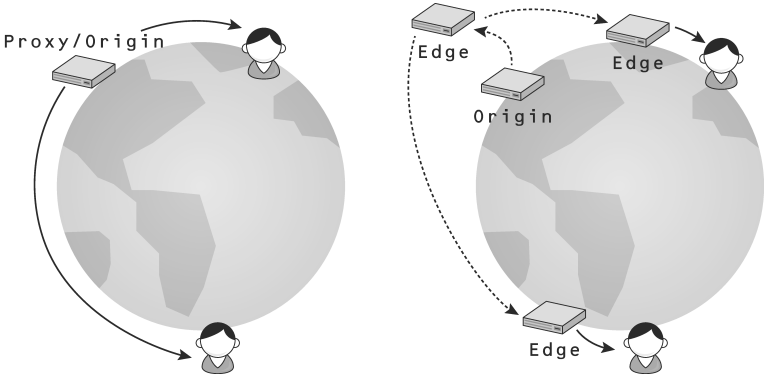

図6.1　CDNのエッジはクライアントの近くにある

もっともわかりやすいCDNのメリットは、配信の高速化、安定化でしょう。

CDNは世界中に高品質のネットワーク、エッジサーバー（クライアントと近い位置にあるサーバー）を有します。そのため、クライアントに地理的に近い位置から、高速かつ安定した配信が可能となります。

CDNをうまく導入すれば、全世界のエッジから、世界中のユーザーに向けて配信がより高速にできます。豊富な回線容量を持つため、大トラフィックもさばけます。

しかし、CDNは入れれば自動的になんとなく配信をいい感じにしてくれるものではありません。さまざまな使い方と注意点があるため、本章では詳細に解説します。

7

自作CDN（DIY-CDN）

自作CDN（DIY-CDN）や自社配信（自作CDNの一部）と外部のCDNを組み合わせるケースについて紹介しました（6.1.2参照）。

自前のCDN（自社CDN）を採用する会社の代表例にNetflixがあります[*1]。莫大なトラフィックが常時ある、配信がコアビジネスであることから、外部のCDNサービスを利用するよりもコストやカスタマイズ性の面でメリットがあるのでしょう。

Netflixのような大規模配信ではなくとも、ほかにも自作CDNがフィットするパターンもあります。そもそも低予算でCDNが使えない、コストを抑えたい、ハイブリッド構築したいパターンです。

本章ではそれぞれのパターンと、実際に低予算で自作CDNをつくるポイントを解説します。

7.1 なぜCDNをつくるのか

CDNを自作するケースは大別すると次の3つです。

- 超大規模サービス。非常に大きなトラフィックや高度なカスタマイズの必要性から、自社構築にコストメリットがある（Netflixなど）
- 低予算。CDNを利用する資金がない
- ハイブリッド構成。CDNと自社配信（自作CDN）のハイブリッド構成によるメリットがある
  - 静的コンテンツのオフロードするためにCDNを使う
  - 定常的に流量があり、自社で低コストで回線調達が可能なため、ベースロードを自社配信として溢れた部分をCDNにオフロードする（DMMなど[*2]）
  - コンテンツを一貫させるために自社配信とCDNの多段構成にする

ほかにもコンテンツの性質上CDNが使えないなど、いくつか例外的なパターンもありえますが、割愛します。上述の3つの中で、多くの読者にとって現実的な

*1 Netflix OpenConnectというサービスを自前で運用。 https://openconnect.netflix.com/ja_jp/

*2 ちなみにAppleも自社でCDNを持っており、ハイブリッド構成となっています。 https://arxiv.org/abs/1810.02978