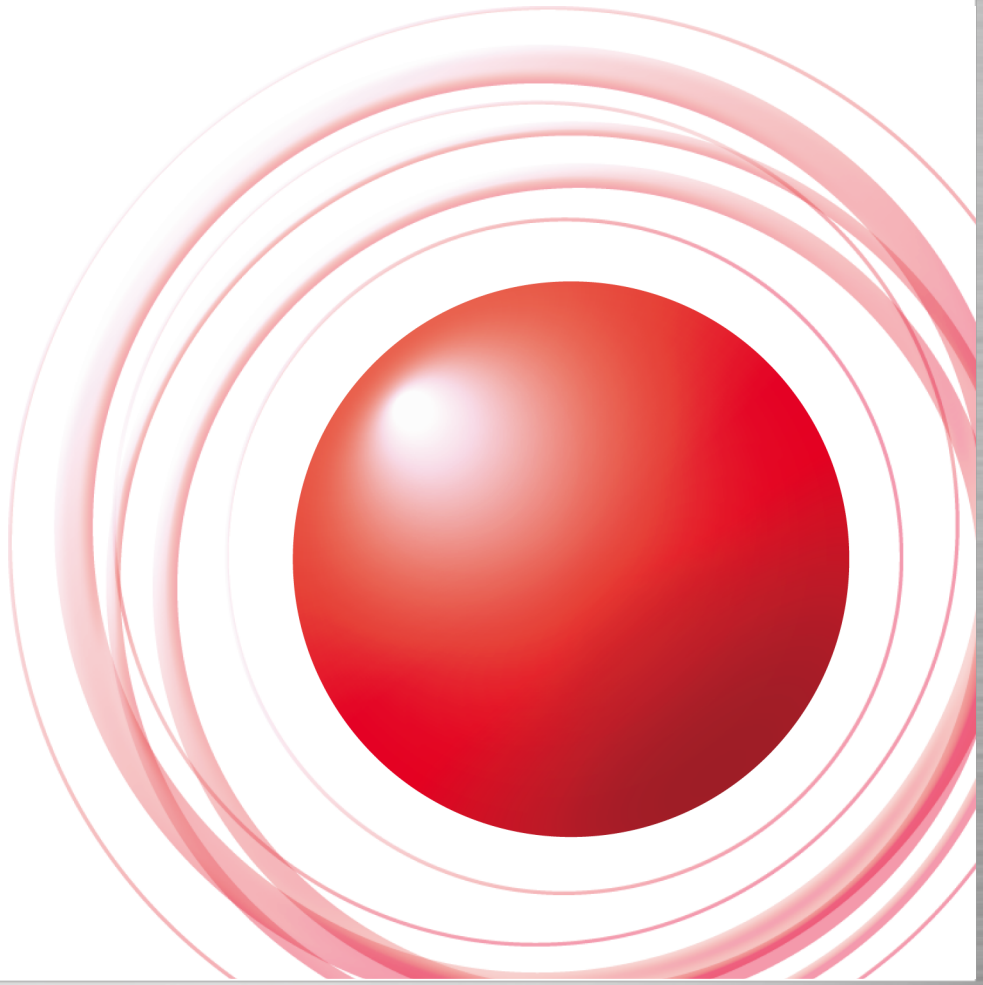
リバースプロキシプロダクトの比較

IIJ Internet Initiative Japan



2012/11/16
株式会社インターネットイニシアティブ
基盤プロダクト開発部 配信技術課
渡辺 道和
Ongoing Innovation

例えば。。。

- Webサイトを運営しているんだけど、サーバの負荷が高くて、泣きそうなんですけど。。。
- でも、同じサーバを並べるのは色々とめんどくさいす。。。

- そんな時こそ、リバースプロキシを立てて、キャッシュさせましょう!!

リバースプロキシを導入する理由

- Webサーバ(オリジンサーバ)の負荷軽減
  - C10K問題
  - 複数のサーバを並べて、スケールさせる
  - オリジンサーバへのアクセスを削減させる
- コンテンツ更新の手間を削減
  - オリジンサーバで保持しているコンテンツを更新し、キャッシュサーバのキャッシュを削除する

代表的なキャッシュプロダクト

よく使われているプロダクト

- Varnish cache
- Apache Traffic Server (ATS)
- Nginx
- IIS Application Request Routing (ARR)

Varnish cache

- フォワード/リバースプロキシとして注目を集めている
- 基本的にオブジェクトはオンメモリで保持
  - 再起動したら忘れちゃいます。。。
- 豊富な管理ツール
  - キャッシュヒット率などをグラフ化するツール
  - 応答時間の分布をグラフ化
  - 最近は稼働状況をjson形式で出力するようになりました

Varnish Cacheの設定(例)

- リクエストの状態遷移に応じて動作を定義
  - VCLという独自言語を使用
- 実際には「コンパイル」して読み込まれる

```
/* default backend */
backend default {
  .host = "origin.example.com";
  .port = "80";
  .host_header = "www.example.com";
}

sub vcl_recv {
  set req.backend = default;
  return(lookup);
}

sub vcl_fetch {
  set beresp.do_stream = true;
  set beresp.ttl = 1h;
  return (deliver);
}

sub vcl_hit{
  if( req.request == "PURGE" && client.ip ~ purge){
    purge;
    error 200 "The request URL is PURGED.";
  } elseif ( req.request == "PURGE" && client.ip !~ purge) {
    error 401 "The request is not allowed";
  }
}

sub vcl_miss{
  if( req.request == "PURGE" && client.ip ~ purge ){
    purge;
    error 404 "The request URL is Not Found.";
  } elseif ( req.request == "PURGE" && client.ip ~ purge ) {
    error 401 "The request is not allowed";
  }
}
```

Varnish Cacheの拡張

- 設定ファイルの中でインラインCを書ける
  - 設定ファイルの中で
    #include <libmemcached/memcached.h>
    とか書けちゃいます
- VModによる拡張
  - 3.0から利用可能
  - インラインCで書かれた部分を外出しするイメージ
- 拡張性、柔軟性
  - VCL < インラインC < VMod

Apache Traffic Server

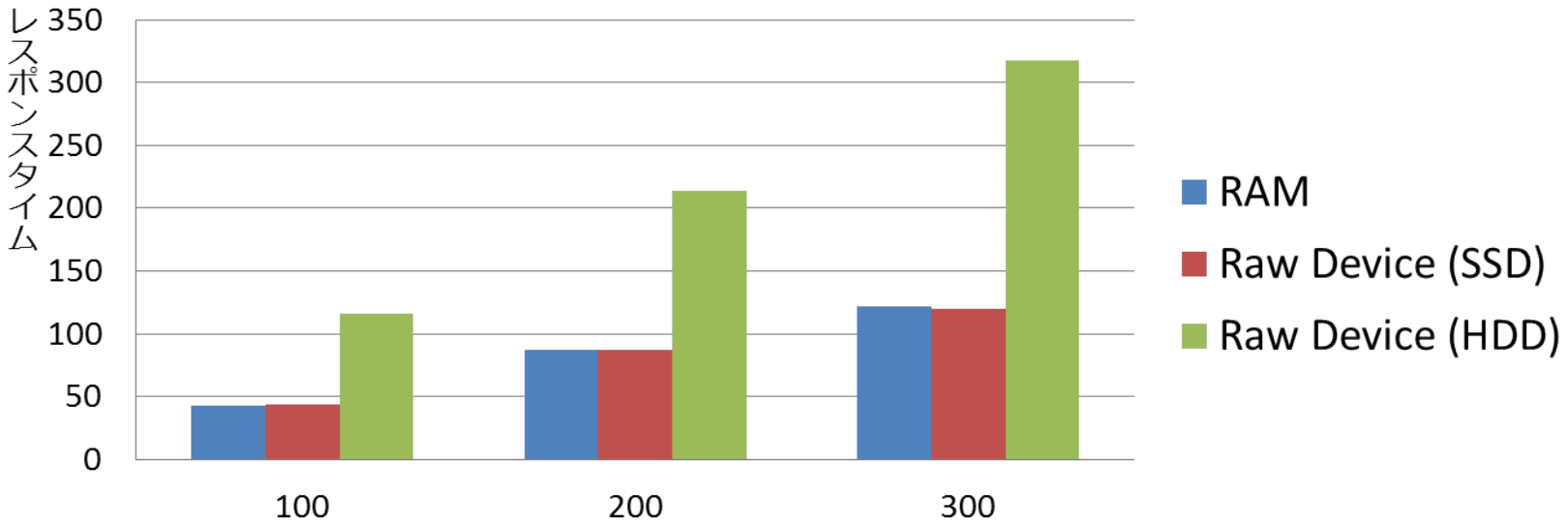
- Inktomiが開発を開始、Yahoo!!を経て Apache Software Foundationに開発が移管
- 大規模サイトで使用したい先進的な機能も

Apache Traffic Serverの特徴

- RAW Device
  - キャッシュ領域として、未フォーマット状態のストレージを使用
  - RAMより安価で大容量のストレージをキャッシュ領域として使えます
  - RAMに乗り切らないサイズのコンテンツのキャッシュも可能
- Split DNS
  - オリジンを指定するときに内部DNSを利用可能
  - 名前解決のオーバヘッドを軽減するためにHost DBという機能が用意されています
- キャッシュの定期アップデート
  - リクエスト状況にかかわらず、定期的にキャッシュをアップデート

Apache Traffic Serverの特徴

(Raw Deviceのパフォーマンス)



- 50Mバイトのコンテンツでキャッシュ領域を変更しながらアクセス
- Raw DeviceをSSDなどDisk I/Oが強いデバイスを使用することでRAM相当の速度が期待できる

Apache Traffic Serverの特徴

- キャッシュ用プロトコルに対応
  - ICP (Internet Cache Protocol)
  - WCCP (Web Cache Communication Protocol)
- クラスタや多段構成にも対応
  - ATS間で設定やキャッシュオブジェクトの共有

Nginxとは?

- webサーバとして最近注目を集めている
  - 大量の接続を処理することができる
  - 軽量なプログラム
  - 純粋なwebサーバとして、シェアを獲得しつつある
- モジュールを組み合わせることでリバースプロキシとしても使用できる

Nginx の特徴

- 豊富な3rd Party module
  - wiki.nginx.orgで多数の3rd Party moduleが紹介されている
  - Github.com でも多くの3rd Party Moduleが登録されている
- 活発な開発活動

Nginx の特徴

- プロセスのオンザフライアップデート
  - サービスダウンを発生させずにバイナリの入れ替えが可能

デモ映像をご覧ください

キャッシュサーバを運用するとき
に気を付ける機能など

キャッシュパージ

- TTLにかかわらず、強制的にキャッシュオブジェクトを削除したい

- それぞれのプロダクトで対応
  - Varnish Cache
    - PURGEメソッドによる削除
    - PURGEリクエストを受け付けた時点では削除されない
    - リクエストを受け付けた時点でパージ対象かを判断する
  - ATS
    - PURGEメソッドによる削除
    - キャッシュインスペクタ(WebUI)からの削除
  - Nginx
    - 3rd Party Moduleにより対応
    - キャッシュ対象を1つづつ指定する必要がある

Request Consolidation

- 同じリクエストが来たときにオリジンに抜けるリクエストを減らす
  - Varnish cacche、ATS、nginxのすべてで対応
  - オリジンの負荷を減らす
- HLS (Http Live Streaming)などでは必須の機能
- 実際にはキャッシュオブジェクトをロックしてリクエストをまとめている

ネガティブキャッシュ

- 404 Not Foundなどのネガティブレスポンスの場合は通常のリクエストと別に扱いたい
- それぞれのプロダクトで対応
  - Varnish Cache
    - オリジンからのレスポンス毎にキャッシュTTLを定義可能
  - ATS
    - あらかじめ決められたレスポンスをネガティブレスポンスとしてTTLを定義できる
  - Nginx
    - オリジンからのレスポンス毎にキャッシュTTLを定義可能

動的コンテンツ

- リクエストに応じて、同一URLでも表示する内容が違う
- 代表的なものとしてクエリストリングなど
  http://www.example.com/foo/bar.html?hoge=huga
- それぞれのプロダクトで対応
  - クエリストリングも含めて全部キャッシュする!!
  - 動的コンテンツを思われるものはキャッシュしない
  - オリジン側でCache-Controlヘッダを付与して、制御する

複数のオリジンサーバを指定

- さすがに1台のオリジンサーバじゃ、心許ない。。。
- それぞれのプロダクトでアプローチが異なる
  - Varnish cache
    - 複数のオリジンサーバが指定されることを前提に実装されている
    - 重み付けによるRound Robin
    - プローブ用のURLを定義
  - ATS
    - 基本的に1つしか定義できない
    - Split DNSを使って頑張る
  - Nginx
    - 重み付けRound-Robinが可能
    - 何かあった時にバックアップサーバを定義可能

SSL

- Name Based Virtual Hostでの悩み
  - SNI (Server Name Indication)
    - ATS / nginxは対応済み
    - TLS拡張の1つ
    - 通信の開始時にクライアント側からサーバ名を宣言
- どこでSSLを終端させるのか?
  - キャッシュサーバ?
  - オリジンまで持っていく?
- サーバ毎にSSLサーバ証明書を用意するコスト
- そもそも、SSL通信の内容をキャッシュするべき?

まとめ

	Varnish Cache	Apache Traffic Server	nginx
キャッシュパージ	○	○	△
Request Consolidation	○	○	○
ネガティブキャッシュ	○	△	○
動的コンテンツ (クエリストリング)	○	○	○
複数のオリジンサーバを指定	○	△	○
SSL	×	○	○

まとめ

- リバースプロキシサーバとして使用できる3つのプロダクト
  - Varnish Cache
  - Apache Traffic Server
  - Nginx
- それぞれのプロダクトで得意、不得意がある
  - 実際には複数のプロダクトを組み合わせて運用
  - リバースプロキシ以外の様々なプロダクトを組み合わせての運用
- ご利用は計画的に!!!
  - キャッシュによる恩恵をうけるためにはコンテンツの設計も重要です

IIJ Internet Initiative Japan

インターネットの先にいます。

IIJはこれまで、日本のインターネットはどうあるべきかを考え、
つねに先駆者として、インターネットの可能性を切り拓いてきました。
インターネットの未来を想い、イノベーションに挑戦し続けることで、世界を塗り変えていく。
それは、これからも変わることのない姿勢です。
IIJの真ん中のIはイニシアティブ ―――― IIJはいつもはじまりであり、未来です。

Ongoing Innovation

お問い合わせ先　IIJインフォメーションセンター
TEL：03-5205-4466（9：30～17：30 土/日/祝日除く）
info@iij.ad.jp
http://www.iij.ad.jp/