internet week 2013

アプリの構造と
そのためのネットワーク

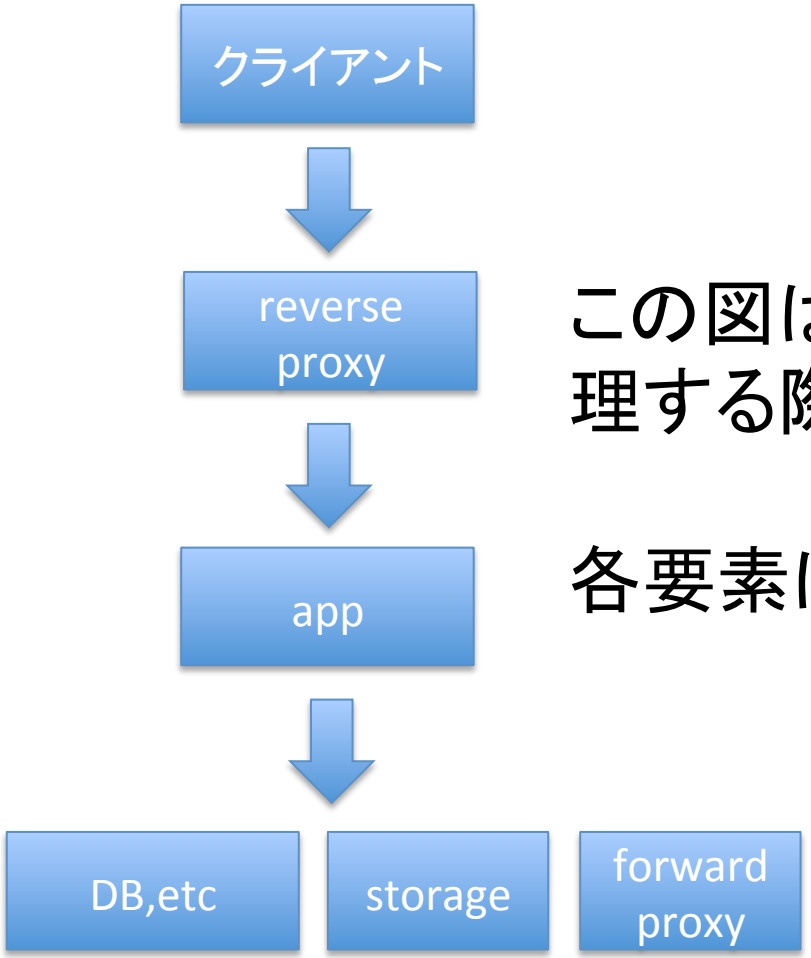
自己紹介

- 吉野純平
- 所属
  - システム本部　運用部　アプリ運用グループ
- 仕事内容
  - トラブル対応
  - コスト計算
  - 等

目次

- コンテンツ生成の仕組み
  - 基礎的な話をします
- アプリの構造による通信の特性
  - コネクションの使い回し
  - パケロスの影響
- 特性を理解した上での実装

アプリケーションの基礎



この図は、リクエストの順序を処理する際の通信の流れの例です

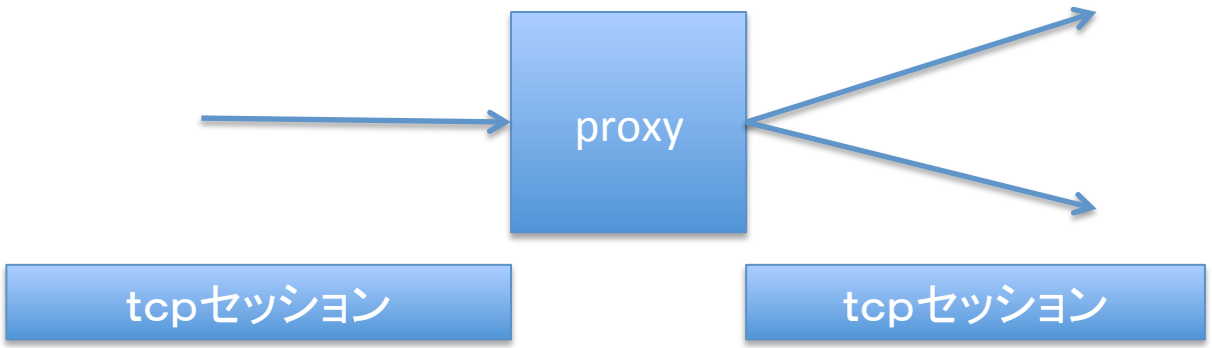
各要素について説明します

app

- アプリケーションのメインの処理を実行する
  - リクエストの内容を理解する
  - 必要な処理を行う
  - 応答内容を作成して送信する
- 方式
  - prefork
    - 接続ごとに前もって作成したプロセスを割り当てる
  - イベント駆動
    - 接続ごとにプロセスを割り当てず、限られた数のスレッドで処理を行う

reverse proxy

- appの前に入れることが多い
- クライアントからの通信をL7で終端する
- 目的の例
  - prefork型appをslowloris系攻撃から守る
  - uri単位で処理するapp分ける
  - appのロードバランシング



db,etc

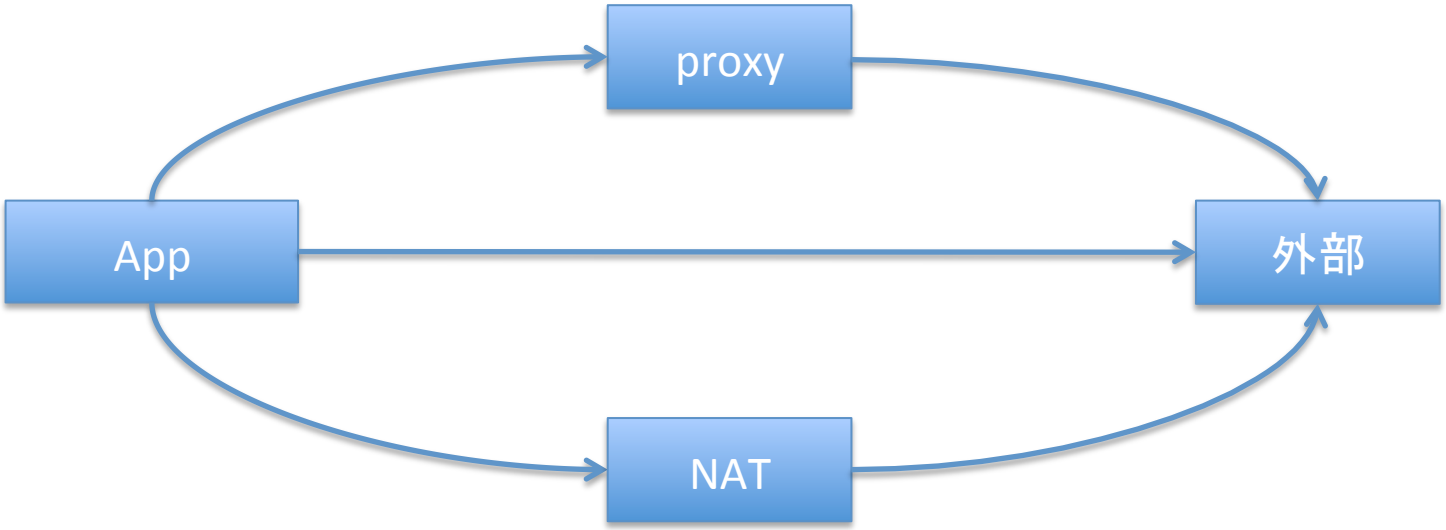
- kvs
  - keyにvalueを紐づけて、活用できる
  - 例）memcached
  - カウンター、キャッシュ等に活用
- RDBMS
  - リレーショナルデータベースマネジメントシステム
  - リレーショナルデータモデルでデータを管理
  - 例）mysql

object data storage

- 画像等のファイルを保管する
- アプリケーションで覆って使う
  - サーバ故障の影響を隠す
  - 故障修理が可能
  - キャッシュ層を持ってサーバ性能を求めない

外部連携の仕組み

- L7で終端するforward proxyを設置する方法
- appに直接グローバルipを設定する方法
- NAT装置でインターネットに抜ける方法



L7終端のpros,cons

- pros
  - アクセス制御が実装されている
    - redirectされたときの扱い
    - domainでのallow deny
    - ip prefixでのallow deny
- cons
  - （実装次第だが）他よりオーバーヘッドは大きい

アプリの構造による通信の特性

tcp接続のコスト

- 接続を維持して、使い回す場合
  - プロセスと接続が1:1の場合、接続数に応じてコンテキストスイッチが増える
  - プロセス等がメモリ等のリソースを消費する
- 毎回新しく接続をする場合
  - 3way hand shakeの時間を待つことになる

ミクシィの場合

- セッション維持コストの低いmemcached
  - プロセスレベルでコネクションを再利用
  - 数万セッションを常に維持している
- セッション維持コストの高いmysql
  - リクエストレベルでコネクションを再利用

パケットロスの影響

- パケロスでサービスの応答速度が落ちる

- 待ちが発生するケース
  - 例）synが落ちる、送ったパケット列が全部落ちる
  - timeout後、再送される
- すぐ再送することで救われるケース
  - 例）dup ackでfast retrans
    - パケットの一部に欠損がある等

特性を理解した上での実装

TCP Retrans Per Sec

会場のみ

どうなってるか推測

会場のみ

対策

- モバイルパケットは落ちる物だと考える
- 落ちても影響をreverse proxyまでに抑えたい
  - appのプロセスを拘束しつづけないようにする
  - app側もリソース解放を適切に行う

センター内のパケロス

- 原因
  - スイッチ故障
  - バッファーサイズが足りないためのtail drop
- 監視
  - スイッチでの監視
  - サーバでの監視

仮想化, SDN

- activeなセッション数はとても多い
- 各種テーブルサイズを抑えるために
  - 仮想化、SDNな未来においてdataplaneにL3以上の情報を入れない
  - できればL2、片方向L3までに制限して考える

まとめ

- アプリのサーバの構成要素を紹介
- 通信特性を紹介
- 仮想化したときに気をつけることを紹介