キャッシュサーバ運用技術

Internet Week ‘99 Tutorial @Yokohama
12/16/99
鍋島　公章
NTT 情報流通プラットフォーム研究所
nabe@slab.ntt.co.jp

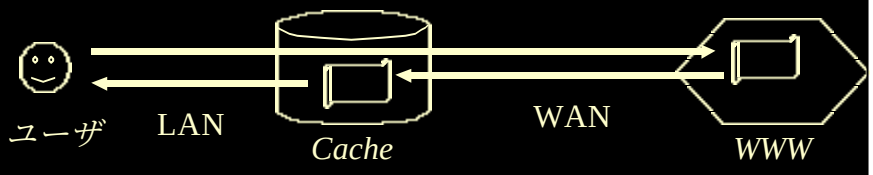
目次

- 概論
- 構築と運用
- ディストリビューションサービス

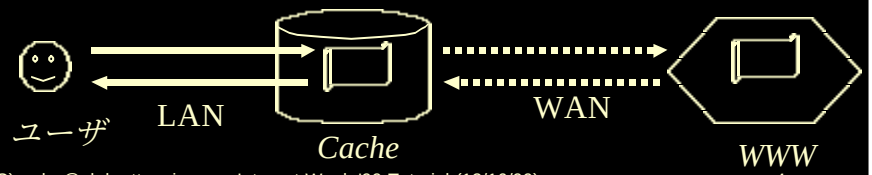
Part 1概論

キャッシュの基本（しくみ）

- リクエストの中継
  - 中継したコンテンツを貯め込む



  - 二回目以降のリクエスト
    - 貯め込んだコンテンツを使う

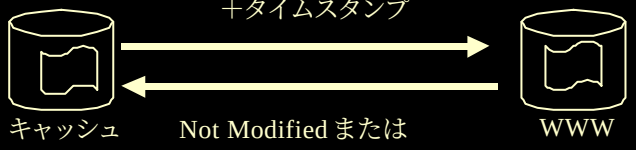


キャッシュの基本（効用）

- 同じコンテンツが，回線上を流れない
  - トラフィック抑制
- 同じリクエストが，WWWサーバに届かない
  - WWWサーバの負荷の低減
- 近くのキャッシュサーバ上にコンテンツがある
  - レスポンスの向上

キャッシュの基本(プロトコル1)

- 最新性のチェック
  - GET IMS (If-Modified-Since)
    - オブジェクトのタイムスタンプ付きGETリクエスト＋タイムスタンプ



- キャッシュ禁止
  - Pragma: no-cache
  - リクエスト
    - 必ずWWWサーバからオブジェクトを取得
  - レスポンス
    - キャッシュに保存禁止

キャッシュの基本(プロトコル2)

- Cache-control (HTTP/1.1)
  - リクエスト
    - キャッシュ許可 (no-cache, no-store)
    - 有効期限 (max-age, max-stale, min-fresh)
    - リクエスト (only-if-cached)
  - レスポンス
    - キャッシュ許可 (public, private, no-cache, no-store)
    - 変更許可 (no-transform)
    - 有効期限 (max-age)
    - 最新性のチェック (must-revalidate, proxy-revalidate)

キャッシュの基本（実際）

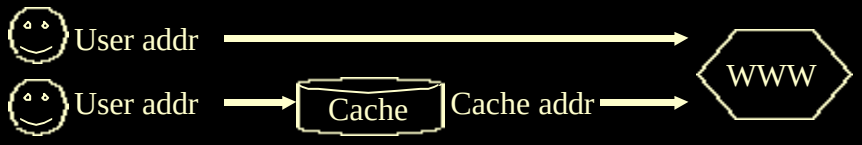
- トラフィック抑制
  - インターラクティブなコンテンツの増加によるヒット率低下
  - 処理トラフィックの増加によるヒット率向上
- レスポンス向上
  - HTTPの平均速度は上がっている
    - 低速PPPユーザだと，キャッシュにヒットしてもレスポンス向上効果が少ない
  - WWW チャット，掲示板サービス
    - リクエストの中継処理の分レスポンスが低下する
- キャッシュサーバの不正利用
  - 外部のユーザに勝手に使われる
  - JPCERT/CCへの報告多数

キャッシュサーバのセキュリティ(1)

- WWWはインタラクティブに
  - もはや，情報を受信するだけではない
  - 商取引
    - WWWショッピング
    - 偽造・盗難クレジットカードの使用
  - コミュニケーション
    - E-Mail，NetNewsの送受信
    - 掲示板，チャットサービス
    - いやがらせ，スパミング，…
  - ポートスキャンの踏台
- 発信者のIPアドレス＝ユーザ特定の重要な方法

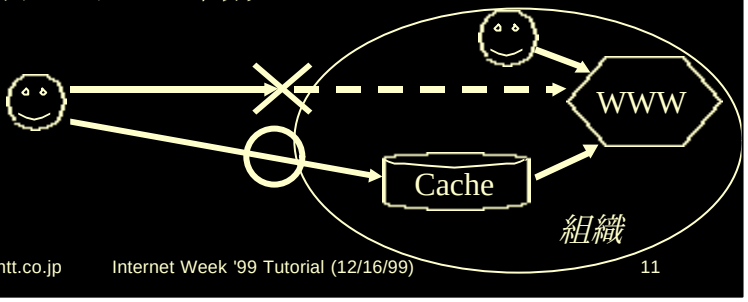
キャッシュサーバのセキュリティ(2)

- 匿名装置
  - キャッシュサーバ
    - ユーザの代理人としてリクエストを発行する
    - サーバ側にはキャッシュサーバのアドレスが残る
  - 匿名装置としてのキャッシュサーバの使用
    - 他の組織のキャッシュを使いプライバシを守る
      - イソターネットの常識？



キャッシュサーバのセキュリティ(3)

- セキュリティホール
  - ファイアーウォールとしてのキャッシュサーバ
  - アクセス制限のミス
    - 外部から内部に入るためのセキュリティホール
    - イントラネット上の社内情報にアクセス可能
  - 特権ポート以外は外部からのアクセスを許すサイトの場合，一台のキャッシュサーバが命取り



キャッシュサーバのセキュリティ(4)

- 外部からアクセス可能なキャッシュサーバを見つけ出すのは簡単
  - ポートスキャン
    - キャッシュサーバ探し専用ツール
  - WWWのアクセスログから特定
    - http_via等によりキャッシュサーバの情報はWWWサーバのログに残る
  - 公開状態のキャッシュサーバのリストも出回る
- ユーザの意識は低い
  - 「インターネットはオープン」という言葉の誤解
    - アクセス可能なキャッシュサーバは誰でも使って良い，と思っている

キャッシュサーバのセキュリティ(5)

- 対策
  - 組織外部からのリクエストを拒否する
    - 一部プロダクトでは，キャッシュサーバでアクセス制限不可
      - ルータでアクセス制限する
    - Apache serverのProxy Moduleも注意
      - 80ポートでProxyが動いている可能性がある
    - Proxy Checker
      - 外部からのアクセス制限のチェック
      - http://cache.jp.apan.net/proxy-checker/
  - アクセスログを保管する

キャッシュサーバのセキュリティ(6)

- 最近のデフォルトのSquid.conf
  - acl all src 0.0.0.0/0.0.0.0
  - acl manager proto cache_object
  - acl localhost src 127.0.0.1/255.255.255.255
  - acl SSL_ports port 443 563
  - acl Safe_ports port 80 21 443 563 70 210 1025-65535
  - acl CONNECT method CONNECT

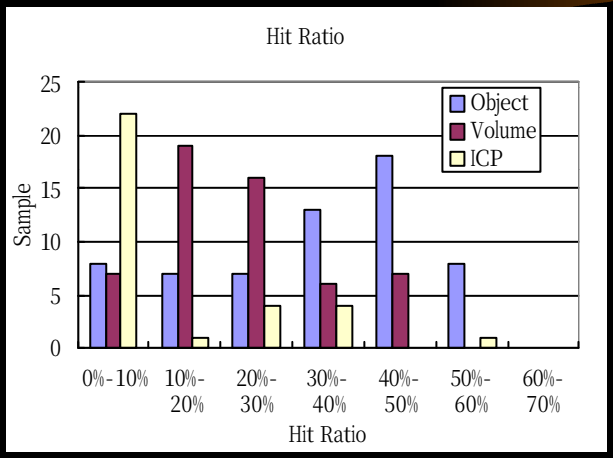
  - http_access allow manager localhost
  - http_access deny manager
  - http_access deny !Safe_ports
  - http_access deny CONNECT !SSL_ports
  - http_access deny all

ヒット率の考察(1)

- アンケート形式のヒット率調査
  - 調査期間
    - 98年11月～12月
  - 報告者数
    - 44
  - キャッシュサーバ数
    - 61
  - http://cache.jp.apan.net/survey98/

ヒット率の考察(2)

- 結果



ヒット率の考察(3)

- ヒット率
  - Object （リクエスト）ベース： 40%程度が中心
  - Volume（トラフィック）ベース：20%程度が中心
  - IIJ4U 36%, 17% (700 Kbps), 某ISP 30%, 30%
- 傾向
  - ユーザの種類，クライアントの種類によってヒット率の傾向は異なる
    - ユーザの種類（企業，学校，ISP）
      - 企業 > 学校 > ISP
    - クライアントの種類（ブラウザ，他のキャッシュ，混合）
      - ブラウザ > 他のキャッシュ

ヒット率の考察(4)

- アンケート結果の内訳
  - 使用ソフトウェアはSquid
    - もはやベストプロダクトではない
  - 最大100万リクエスト/日（数M bps程度）
- 最近の製品を使うとヒット率は高くなる
  - 処理トラフィック（処理トラフィックが増えるとヒット率は向上する）
    - 10Mbps以上の処理が可能
  - ヒット率を高める機構

ヒット率の考察(5)

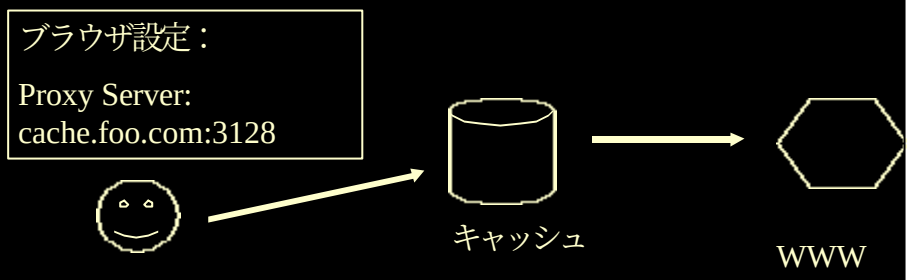
- ヒット率を高める機構
  - アルゴリズムの改良
    - コンテンツの更新処理
    - ユーザのReloadリクエストへの対応
  - コンテンツの自動更新
    - 人気コンテンツの定期的アップデート
      - Mirror-Image, CacheFlow, Microsoft

新しい運用形態

- 透過型キャッシュ
- リバースプロキシ
- 商用キャッシュサービス

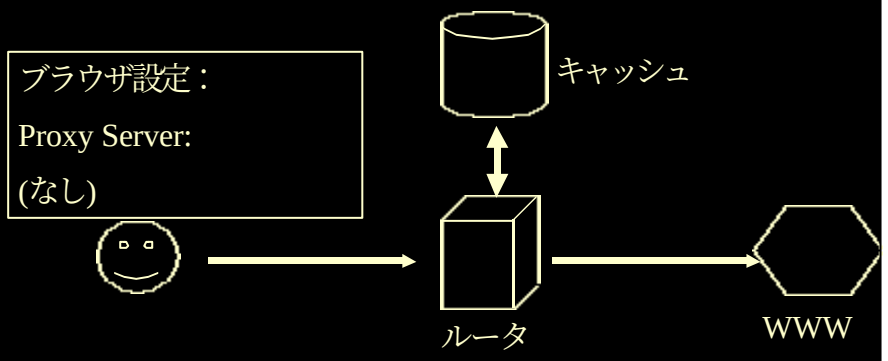
透過型キャッシュ (しくみ1)

- Proxy型(従来型)
  - ユーザは，明示的にキャッシュサーバを設定



透過型キャッシュ(しくみ2)

- 透過型
  - ルータ等が強制的に80番向けパケットをキャッシュに送り込む（リクエストのハイジャック）



透過型キャッシュ（利点）

- ユーザの設定が不必要
- 全トラフィックをキャッシュ可能
  - トラフィック抑制効果が大きい
    - 通常だと，使わせ方に悩む
- トラフィックを落としたい所だけ，キャッシュが使える
  - 経路に沿って配置可能
  - 例
    - 国際線（リンクコスト高い）はキャッシュする
    - 国内線（リンクコスト安い）はキャッシュしない

透過型キャッシュ（問題点）

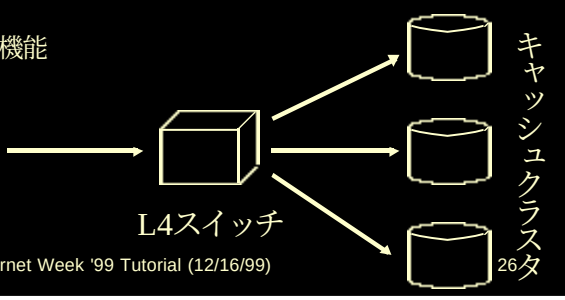
- ユーザは強制的にキャッシュを使用させられる
  - 安定性，プライバシがより重要に
- 特別なルーティング
  - ルータの負荷が上がる
- 強制的に多段キャッシュとなる可能性
  - 最適な運用形態
- 80番ポートを使うWWW以外のアプリケーション
  - 世の中には変なシステムが存在する

透過型キャッシュ（現状）

- 米国
  - 結構使われている（特にISP）
    - L4スイッチとのセット
- 日本
  - ユーザサイトでは，ぼちぼち導入済み
  - ISPによる使用も始まった
    - 大手ISPも導入直前

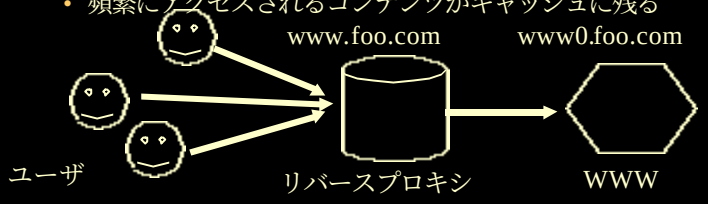
透過型キャッシュ(L4スイッチ)

- Layer4 (TCP) コネクションの振り分け・変換
- 最近のキャッシュサーバは，単体でも透過型が可能
  - 比較的大規模な構成ではL4スイッチの使用が主流
- L4スイッチの機能
  - クラスタリング
    - ハッシュを使ったコンテンツの分散
  - フェイルオーバー
  - 透過型キャッシュ機能



リバースプロキシ（基本1）

- 特定のWWWサーバ用のキャッシュ
  - WWWサーバの代理にリクエストを受け付ける
    - 頻繁にアクセスされるコンテンツがキャッシュに残る



  - 役割
    - WWWサーバの高速化
    - 軽量ミラーサーバ

リバースプロキシ（高速化）

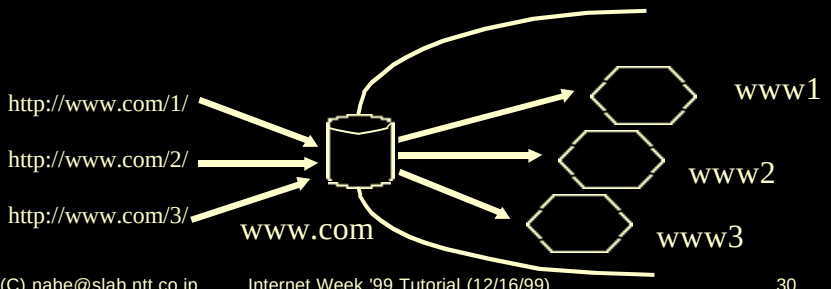
- WWWサーバの高速化
  - WWWサーバでも同様の機構を実現可能だが，現状では，リバースプロキシを使うのが一般的
  - WWWサーバの一つのボトルネックはディスク処理
    - 人気コンテンツをメモリ上に持つ
      - ディスクIO処理の回数が減る
      - 人気コンテンツへのリクエストが高速化される
  - リクエスト処理の高速化
    - シングルスレッド等
  - 動的なコンテンツ（データベースアクセスCGI等）のキャッシュ
    - システム設計に注意が必要
    - サーバ負荷を大幅に低減

リバースプロキシ（軽量ミラーサーバ）

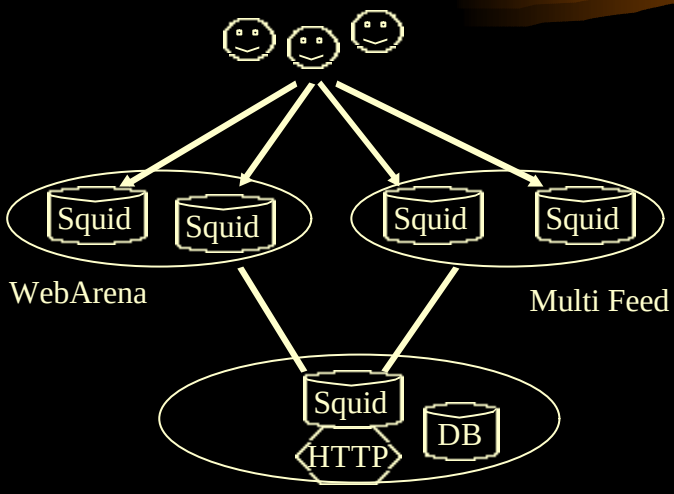
- 一種のミラーサーバ
  - 頻繁にアクセスされるコンテンツだけが，リバースプロキシ上にキャッシュされる
    - 頻繁にリクエストされるコンテンツは全体の一部
      - 必要なディスク領域の節約
  - WWWサーバからの情報の更新は必要ない
    - コンテンツはユーザのリクエストにより自動的にキャッシュ上に貯まる
    - ユーザのReloadリクエスト等により自動的に更新される
  - 発展形として，WWWサーバから更新情報をリバースプロキシにプッシュする機構もある
    - ディストリビューションサービス

リバースプロキシ（その他用途）

- 内部サーバの保護
  - ファイアーウォール上にリバースプロキシを配置
    - 外部からのHTTPリクエストだけを，内部のWWWサーバに中継
- 複数のサーバを一台にまとめる
  - URL書換え



リバースプロキシ (例)



WWW以外のアプリケーションへの適用

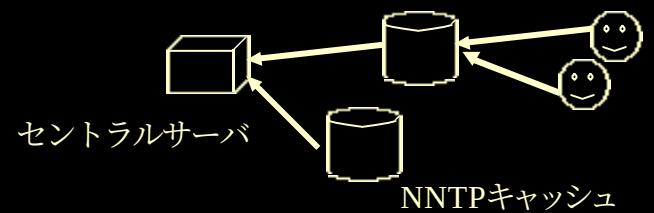
- NEWS
- Stream

NEWS(1)

- 既存のUsenetのモデル
  - グループ単位の配送
    - 実際にユーザが読むかどうか？
  - それぞれのサーバ間で記事のバケツリレー
    - サーバ単位で記事やNews Groupの管理
      - 管理コスト，処理コストが大きい

NEWS(2)

- NNTPキャッシュの構成
  - セントラルサーバ
    - 既存のNewsサーバ
      - 記事やNews Groupの管理
      - 記事の配送
    - ユーザからのリクエストは直接受けない
  - NNTPキャッシュサーバ
    - 読まれた記事のキャッシュ
      - キャッシュコンテンツの管理
    - News Groupの管理は自動



NEWS(3)

- メリット
  - ディスク容量，ネットワークトラフィックの節約
  - 管理コストの低減
- 運用例：
  - 複数のNewsサーバを管理
    - 一台をセントラルサーバ，その他をNNTPキャッシュサーバ
    - NNTPキャッシュサーバ間のラウンドロビンも可能
  - 小規模なNewsサーバを管理
    - 他の組織のセントラルサーバを使い，ローカルにはNNTPキャッシュサーバのみ

NEWS(4)

- プロダクト
  - Inktomi Traffic Server
    - 1999年リリース
    - 記事の保管に高性能ファイルシステムを使用
      - 高いパフォーマンス
    - 安いソフトではない
  - Delegate
    - 数年前から実装されている
    - 運用実績もそこそこ
    - フリーソフト
  - NNTP Cache
  - News Cache
  - Entra

Stream(1)

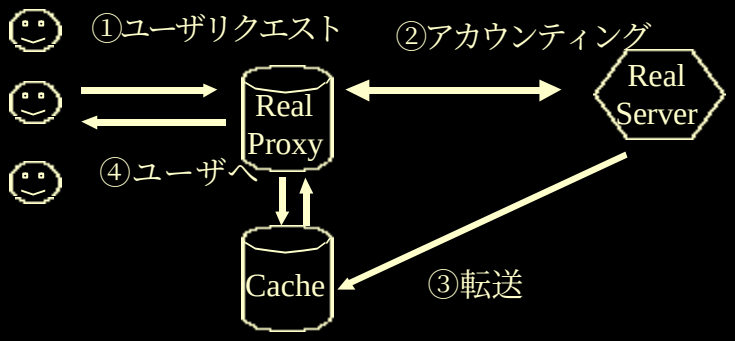
- 21世紀のメイントラフィック
  - Webトラフィックを超えると見られている
    - TV程度の品質=1.5 M bpsの帯域が必要
- ネットワークの安定性&太さが重要
  - コンテンツをユーザの近くに置く(複製）効果が大きい
- キャッシュ技術の次のターゲットはストリーム
- プロダクト
  - Inktomi, Network Appliance, InfoLibria, Entra

Stream(2)

- Real Networks用キャッシュ
  - 特徴
    - アカウンティング
      - キャッシュ上のコンテンツもアカウンティングを行う
        - 配布はセントラルサーバで管理
    - サーバからキャッシュへの転送は特別のプロトコルを使用
      - オブジェクトの完全な転送を行うため
    - キャッシュサーバは一種のファイルシステムとして使う
      - 古いオブジェクトの自動的消去

Stream(3)

- 概要



プロダクト

- 選択の基準
  - 安定性，保守性
    - 最も重要
    - プロダクトが安定するには時間がかかる
  - 性能，値段
    - 徐々に差が小さくなっている

ベンダ

- 御三家
  - Inktomi
  - Network Appliance
  - CacheFlow
- 新興
  - Novell
  - Dell
  - InfoLibria
  - Entra

ベンチマーク (1)

- WWWサーバ用のベンチマークは使えない
  - 小数のデータに対するリクエスト
    - リクエストの局所性
      - ヒット率が高くなり，あまりディスクを使わない
- キャッシュサーバ用のベンチマーク
  - 適度なヒット率を保つリクエストパターンを生成
- Web Polygraph (IRCACHE)
  - 複数のクライアントとサーバ
  - 定期的にベンチマーク大会を開催

ベンチマーク(2)

- IRCACHE bakeoff (March/99)
  - プロダクト　req/sec
  - InfoLibria (large)　1680 (Cluster:4)
  - InfoLibria (small)　690
  - Novell/Dell (large)　1500
  - Novell/Dell (small)　400
  - Peregrine　620
  - Squid　96
  - 注:
    - 10req/sec = 1 M bps (ATRCの運用データ)

ベンチマーク(3)

- DataCommunication (July/99)

プロダクト	price($)	req/sec
CacheFlow (CacheFlow 5000)	191,990	2000 (Cluster:2)
Cobalt (Cacheraq 2)	2,599	30
Eolian (InfoStorm RA2)	9,200	70
Dell (Novell ICS by Dell)	69,820	2300
Novell (Novell ICS)	293,967	6000 (Cluster:3)
NetCache (Netcache C720)	23,125	440
NetHawk (Nethawk/1.0)	8,300	540

Part 2

- 構築と運用

運用心得

- ユーザリクエストの中継
  - ルータの管理と同じ
- プライバシーの保護
  - ユーザのプライバシ(アクセス披歴)を握る

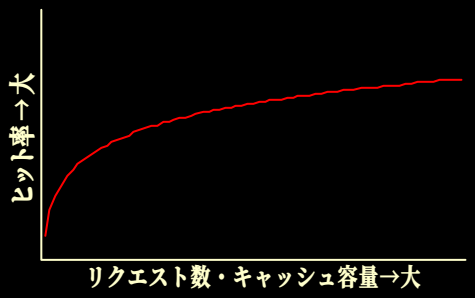
運用トレンド

- アクセス制限は必須
  - 悪用の広がり
- キャッシュサーバの協調運用は難しい
  - 協調関係は最小にする
- サーバ負荷に注意
  - 高負荷のキャッシュはレイテンシを増加させる
- フリーソフト以外のプロダクトが低価格化
  - ある程度以上のトラフィックになるとSquidは管理コストが高い

協調運用（ヒット率の上げ方）

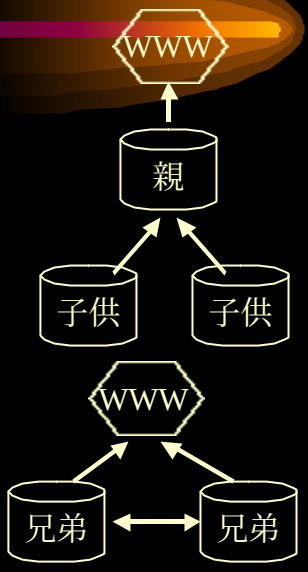
- キャッシュが処理するリクエストを増やす
- キャッシュ領域を増やす

- ただし，対数に比例



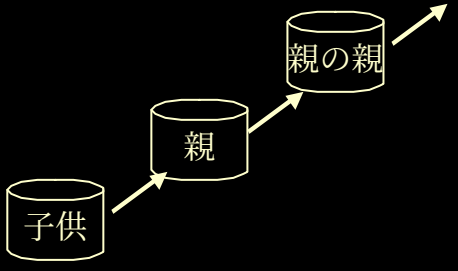
協調運用(関係)

- 親子関係
  - 子供はミスした全リクエストを親に送る
    - 親キャッシュで処理するリクエストが増加
    - →親キャッシュのヒット率が上がる
    - →系全体のヒット率が上がる
- 兄弟関係
  - コンテンツがある場合のみ，リクエストを送る
    - 見かけ上のキャッシュ領域が増加
    - →ヒット率が上がる



協調運用（親子関係）

- 多段接続は避ける
  - １つのキャッシュを経由する毎に0.5秒程度＋ICP待ち時間（キャッシュ間のRTT+ICP処理時間）の時間が余計にかかる
  - 3段程度／約2秒が限度
- 速い経路を持つキャッシュが有効
  - 親キャッシュの経路が使用される
- ICPを使うのが無難
  - 親のダウンを検出

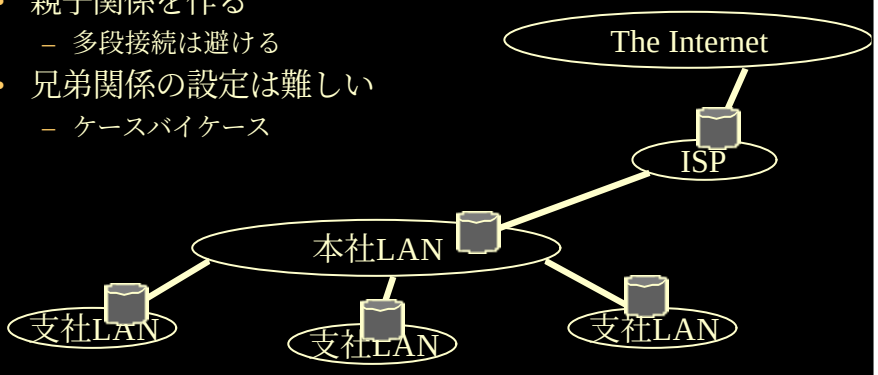


協調運用（兄弟関係）

- 下手な協調関係はレスポンスを低下させる
  - 効果のある兄弟サーバを選ぶ
    - RTTが小さい（最重要）
      - 一番大きいRTTに引きずられる
    - 太い線で接続されている
    - ヒット率が高い

協調運用（キャッシュの配置）

- 対外リンクの手前に１システムが基本
  - サイト（同一LAN内）に１システム
  - 組織単位等では置かない
- 親子関係を作る
  - 多段接続は避ける
- 兄弟関係の設定は難しい
  - ケースバイケース



Squid + FreeBSD

- ターゲット
  - 1M bps程度まで,
    - 使い古したPC+FreeBSD+Squidで十分対応可能
      - 256 MB程度のメモリは載せる
  - 3M bps程度
    - 30万円程度のPC
      - 512MB程度のメモリ
      - ディスクに投資
  - 5M bps以上
    - 単体のSquid + FreeBSDでは荷が重い
    - クラスタリング

ハードウェアの選定(1)

- ディスク性能が一番効く
  - 高性能ディスクを複数台接続
    - シーク待ち時間を減らす
    - 6台程度までは向上する
      - あまりに多いとSCSIが飽和する
- できるだけ多くのメモリをのせる
  - 20GBのオブジェクト=1M個程度のオブジェクト
    - オブジェクトのインデックスサイズ=50MB程度
  - オンメモリキャッシュ領域として使う
    - ディスクIOを減らす
- CPUはそれほど重要ではない
  - そこそこのCPUで十分
  - Squidの場合，Multi CPUは意味がない

OSのインストール(パーティション)

- パーティション
  - キャッシュ領域は別パーティションにする
    - noatime オプション付きでマウント
      - アクセス時間を書き込まない
  - LOG領域は大き目に取る
  - スワップ領域を大きくしても意味がない
    - スワップを使い始めるとSquidの性能は著しく低下する

OSのインストール(Log領域1)

- Logファイルの大きさ
  - access.log
    - ICP + TCPリクエスト
    - 130 byte / 行　程度
    - gzip -9
      - 1/5程度に圧縮可能
    - 例：　10万リクエスト
      - 生log:　　13M
      - 圧縮log: 2.5M
  - store.log
    - オブジェクトのディスクIOのlog
      - 通常，不必要
    - access.logと同程度
  - cache.log
    - エラーメッセージ
    - 通常，無視出来る大きさ
    - エラー時に増えるが大したことはない

OSのインストール(Log領域2)

- Logファイル用領域
  - 一日に必要な量の最低5倍程度の領域を確保する
    - logの保存失敗
    - 急なリクエストの増加
  - 例：10万リクエスト/日（access.logのみ）
    - store.logを残す場合，同程度の領域がさらに必要
    - 10万行　* 130 = 約 13 MB/日
    - Log領域:　　60 MB程度/日を用意する
    - 保存用:　　2.5 MB程度/日

OSのインストール（時計）

- 時計あわせ
  - オブジェクトの新鮮さの判断
    - （オブジェクトの取得時刻ーオブジェクトの生成時刻）＊α
    - 時計の遅れ
      - オブジェクトがキャッシュされない
        - 未来のファイルをキャッシュ
    - 時計の進み
      - 古いオブジェクトを返す
  - NTP等を設定する

OSのインストール（カーネル1）

- 使用可能リソースを増やす
  - プロセスあたりの使用可能ファイルディスクリプタ数
    - MAXFILES
      - デフォルト 1064
    - リクエストの中継には3つのファイルディスクリプタを使用
      - 同時に300コネクション程度しか処理できない
      - ディスクリプタを使い切ると，レスポンスが急激に低下
        - コネクション開始の待ち行列に入る
  - MBUF領域
    - NMBCLUSTERS
      - デフォルト 1024
    - 処理するコネクション数が増えると間に合わない
      - OSが落ちる

OSのインストール（カーネル2）

- カーネルのコンフィギュレーションファイルの編集
  - cp /sys/i386/conf/GENERIC CACHE
    - maxusers 32 -> 256
      - options MAXFILES = 2048
    - options NMBCLUSTERS = 10240
- カーネルのコンパイル
  - config CACHE
  - cd ../../compile/CACHE
  - make clean
  - make depend
  - make
  - cp kernel /kernel

コンパイル

- コンパイル
  - tar -xzf squid-2.2.STABLE5-src.tar.gz
  - cd squid-2.2.STABLE5
    - ./configure
    - make
    - make install

Squid.conf（アクセス制限 1）

- 最近のデフォルトのSquid.conf
  - acl all src 0.0.0.0/0.0.0.0
  - acl manager proto cache_object
  - acl localhost src 127.0.0.1/255.255.255.255
  - acl SSL_ports port 443 563
  - acl Safe_ports port 80 21 443 563 70 210 1025-65535
  - acl CONNECT method CONNECT

  - http_access allow manager localhost
  - http_access deny manager
  - http_access deny !Safe_ports
  - http_access deny CONNECT !SSL_ports
  - http_access deny all

Squid.conf（アクセス制限2）

- クラス設定
  - acl users src 127.0.0.1/255.255.255 123.456.0.0/255.255.0.0
- リクエスト制限
  - http_access deny !users
- コメントアウト可能
  - http_access deny !Safe_ports
    - ポートスキャン，不正利用の禁止

Squid.conf（キャッシュ領域）

- ディスクキャッシュの領域の設定
  - 実際のパーティションの大きさの8割程度を割振る
  - Cache_dir /usr/local/squid/cache 800 16 256
- メモリキャッシュの領域の設定
  - スワップを使わない程度に大きくする
    - 一般には実メモリの1/3程度
      - 例：ATRC 実メモリ 512MB キャッシュメモリ 192MB
        - 100MB程度のFreeメモリ
  - cache_mem 80 MB
- 最大オブジェクトサイズ
  - 例：maximum_object_size 128000

Squid.conf（その他必須項目）

- Mailアドレス（Mail aliasも設定する）
  - ftp_user squid@your.cache.com
  - cache_mgr root@your.cache.com
- DNSキャッシュクライアント数
  - dns_children 5 -> 32

Squid.conf（ヒント1）

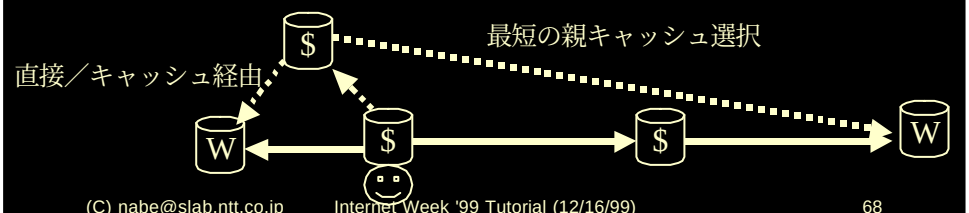
- 負荷を下げる
  - abortしたリクエストを強制終了させる
    - quick_abort_(min|max|pct)
  - 最大オブジェクトサイズを小さくする
    - リクエストあたりのディスクIOを減らす
    - maximum_object_size
- ヒット率を上げる
  - オブジェクトの新鮮さの判断を甘くする
    - refresh_pattern
  - 強制ReloadをIMSに書換え
    - reload_into_ims on

Squid.conf（ヒント2）

- セキュリティ
  - リクエストの中継情報を非表示
    - 内部ホストの情報を外部に出さない
    - forwarded_for off
- ファイアーウォール内キャッシュ
  - 必ずファイアーウォール経由でアクセス
    - acl local dstdomain your.com
    - always_direct allow local
    - never_direct allow all
    - cache_peer no-query

Squid.conf（ヒント3）

- 最短経路の選択
  - --enable-icmp（コンパイル時）
  - query_icmp on
  - NetDB
    - WWWサーバまでのRTT
    - リクエスト時にICMPをWWWサーバに送る
    - RTTの足し込み（親キャッシュ経由）



実行準備

- ディレクトリ設定
  - Ownerをnobodyに変更
    - /usr/local/squid/log
    - /usr/local/squid/cache
  - キャッシュ用領域の初期化
    - /usr/local/squid/bin/squid -z
- 実行ファイル
  - /usr/local/etc/rc.d/squid.sh
    - /usr/local/squid/bin/RunCache &

アクセスLog(解析，保存)

- Calamari
  - オプション
    - -a 詳細な表示
    - -r -1 全クライアント情報表示
  - スクリプト
    - $SQUID/bin/squid -k rotate
    - sleep 120
    - mv $SQUID/log/cache.log.0 $ARC/cache.$DAY
    - calamari < $ARC/cache.$DAY > $STAT/stat.$DAY

アクセスLog(フォーマット)

- フォーマット
  時刻　転送時間　クライアントアドレス
  キャッシュスステイタス/httpステイタス　オブジェクトサイズ
  HTTPメソッド　URL　ユーザ名(ident)
  協調ステイタス/協調ホスト mime

  942289905.074　537 129.60.215.113
  TCP_MISS/200 1949
  GET　http://foo.com/index.html -
  FIRST_PARENT_MISS/cache00.jp.apan.net text/html

アクセスLog(キャッシュステイタス1)

- 基本
  - TCP_HIT
    - 通常のヒット
  - TCP_MISS
    - 通常のミス
  - TCP_MEM_HIT
    - オンメモリキャッシュにヒット
  - TCP_DENIED
    - アクセス制限違反
  - TCP_NEGATIVE_HIT
    - エラーステイタスのキャッシュにヒット
      - HTTP not found (404)等
  - TCP_SWAPFAIL
    - ヒットしたが，実際にはキャッシュ中に存在しない（システムエラー）
  - ERR_CLIENT_ABORT
    - クライアントがリクエストを中断
  - ERR_NO_CLIENTS
    - クライアントが強制終了
  - ERR_READ_ERROR
    - WWWへのリクエストの読み込みエラー
  - ERR_CONNECT_FAIL
    - WWWへコネクションが開けない

アクセスLog(キャッシュステイタス2)

- Squidによる最新性のチェック(IMSリクエストの発行)
  - キャッシュにヒット，しかしオブジェクトが長期間キャッシュに存在
    - （オブジェクト取得時間ーオブジェクト生成時間）＊α
  - TCP_REFRESH_HIT
    - キャッシュ中オブジェクトは最新
      - キャッシュ中のオブジェクトを返す
  - TCP_REFRESH_MIS
    - オブジェクトをWWWサーバから取得
      - 最新のオブジェクトを返す
  - TCP_REF_FAIL_HIT
    - IMSリクエスト失敗
      - キャッシュ中のオブジェクトを返す

アクセスLog(キャッシュステイタス3)

- クライアントによる最新性のチェック
  - TCP_CLIENT_REFRESH
    - クライアントがno-cache命令を発行
      - 常にWWWからオブジェクトを取得
  - TCP_IMS_HIT
    - クライアントがIMS命令を発行，しかし，キャッシュ中のオブジェクトは最新
      - Not Modifiedを返す
  - TCP_IMS_MISS
    - クライアントがIMS命令を発行，キャッシュ中のオブジェクトが古い事が判明
      - 最新のオブジェクトを返す

アクセスLog(協調テイタス1)

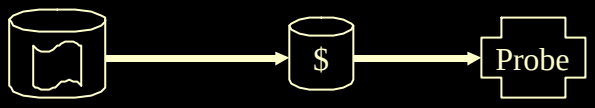
- 基本
  - NONE
    - キャッシュにヒット（外部アクセスなし）
  - DIRECT
    - WWWサーバに直接アクセス
  - TIMEOUT_DIRECT
    - ICPのタイムアウトによりWWWサーバに直接アクセス
  - PARENT_HIT
    - 親キャッシュでヒット
  - PARENT_MISS
    - 親キャッシュでミス
  - SIBLING_HIT
    - 兄弟キャッシュでヒット

アクセスLog(協調テイタス2)

- 最短経路
  - FIRST_PARENT_MISS
  - FIRST_UP_PARENT
  - SOURCE_FASTEST
  - CLOSEST_DIRECT
  - CLOSEST_PARENT_MISS
  - CLOSEST_PARENT_MISS
- その他
  - DEFAULT_PARENT
  - SINGLE_PARENT
  - ROUNDROBIN_PARENT

監視(1)

- リソース不足
  - リクエストの中継時間が長くなる
  - 監視
    - 特定のファイルを取り出す時間を計測する
  - pl-checker (proxy latency checker)
    - 中継速度の計測
    - timeoutすると管理者にMailで連絡
    - 例：pl-checker proxy-server 3128 http://target-url 10 nabe@slab.ntt.co.jp



監視(2)

- Multi Router Traffic Grapher (MRTG)
  - 本来は，Routerの処理トラフィックを表示するWWWページ作成ツール
  - Squidにも使える

ユーザへの使わせ方 (1)

- キャッシュサーバを積極に使うユーザの割合は減少傾向
  - キャッシュを使っても，あまりレスポンスは上がらない
    - 特に低速ＰＰＰユーザ
    - Web Chatや掲示板ではレスポンスが下がる
  - 第三者にリクエストのＬＯＧを管理される
    - 嫌がるユーザも多い

ユーザへの使わせ方 (2)

- 強制的に使わせる
  - 80ポートへのリクエストをブロックする
    - キャッシュサーバを使わないと，外部のWWWへアクセス不可能
    - 大学等では行われている
  - 透過型キャッシュ
    - FreeBSD単体
      - IF Filter + Squid ipf-transparentオプション
    - Cisco
      - WCCP 1.0
      - Port redirection

ユーザへの使わせ方 (3)

- PAC (Proxy Auto Config)
  - プロキシサーバの自動設定
  - proxyサーバの情報をJavascriptで記述
    - ドメイン，IPアドレス等によるプロキシサーバの選択
    - バックアップサーバの指定

ユーザへの使わせ方 (4)

- WPAD (Web Proxy Auto-Discovery protocol)
  - PACファイルのあるURLの自動認識
  - 使用プロトコル
    - Dynamic Host Configuration Protocol (DHCP)
    - Service Location Protocol (SLP)
    - Well-known ホスト名(WPAD)
      - http://wpad.your.domain/wpad.dat
    - DNS レコード (SRV TXT)
  - IE5.0で使用可能
    - LANの設定オプション

協調運用

- 基本設定
  - 親子
    - cache_peer 親キャッシュ parent TCP-port ICP-port
  - 兄弟
    - cache_peer 兄弟キャッシュ sibling TCP-port ICP-port
- 詳細設定
  - cache_peer_domain
  - cache_peer_access
  - neighbor_type_domain

クラスタリング

- 一台のサーバではパフォーマンス不足
  - マシンを増やすのが低コスト
    - 2台にすれば，処理性能は2倍
  - ラウンドロビン
    - DNS
    - PACファイルの利用
  - コンテンツの分散を行うとヒット率の向上が見込める
    - ハッシュによるコンテンツの分散
    - ハッシュを使わないコンテンツの分散

ハッシュによるコンテンツの分散

- クライアントーサーバ間
  - PACファイルの利用
    - Super Proxy Script
      - proxyIndex = Mod (CheckSum(URL), N)
  - L4スイッチの利用
    - 仮想サーバの設定
    - リクエストの振り分け
- サーバーサーバ間
  - CARP (Cache Array Routing Protocol)
    - ハッシュ関数によるURLのグループ分け
      - キャッシュサーバ間のコンテンツの分散
    - Proxy Array Membership Table
      - キャッシュサーバのグループの定義

ハッシュによらないコンテンツの分散

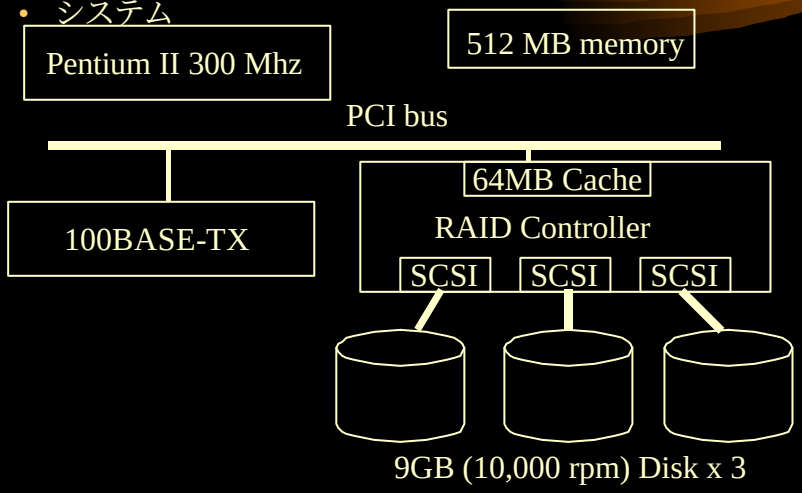
- キャッシュはラウンドロビンでリクエストを受付ける
  - 相互に兄弟関係かつProxy-only オプション
    - 他のキャッシュから得たオブジェクトはローカルに保存しない
    - 大体，分散できる
  - Cache Digest
    - 定期的にオブジェクトの一覧を交換する
    - かなりラフな分散になる
  - ICPの利用
    - ICPパケットの量は無視できない
    - Multicast ICP

実システム(1)

- APAN Tokyo Root Cache Server (ATRC)
  - APAN (Asia-Pacific Advanced Network)
  - 大学，政府研究機関がユーザ
    - 約2 0サイト
  - Root サーバ
    - 各サイトのキャッシュサーバからのリクエストを受ける
  - ハードウェアスペック
    - Pentium II 300MHz，512 MB Memory
    - 9.1GB Disk x 3 (Ultra-Wide 10,000 rpm)
    - RAID Controller (PCI to SCSI)
      - DPT PM3343UW/3+64MB Cache
    - 100BASE-TX

実システム(2)

- システム



実システム(3)

- 設定
  - ディスク全体をRAID-0 (ストライピング)
    - ３台のディスクを最大限に使う
    - 一台のディスクが故障するとシステム全体の再インストールが必要（故障する確率は単体のディスクの３倍）
- 運用状況
  - １００万リクエスト/Day
  - １０GB/Day
  - ヒット率(リクエストベース): 20%
  - ヒット率(トラフィックベース): 10%
  - ヒット率(ICPベース):10%

実システム(4)

- パフォーマンス
  - 通常，スワップは不使用
  - ピーク時
    - 100リクエスト/Sec (瞬間最大：Calamariによる統計)
    - 30リクエスト/Sec (5分間の平均)
    - 3M bps
    - CPU Usage: 50%
    - 同時TCPコネクション: 500
      - 使用ファイルディスクリプタ： 600
    - たまにMBUF領域を使い切りOSダウン

Part 3

- ディストリビューションサービス

ディストリビューションサービスとは

- 目的
  - ユーザリクエストのレスポンス向上
    - 帯域の節約
    - 巨大コンテンツの配送
  - リライアビリティ向上
    - サーバ故障，ルーティング障害対策
- 主要技術
  - 情報（複製）の最適配置
  - ユーザリクエストのナビゲーション
- ビジネス
  - 情報の配送サービス
  - 一種のミラーサーバ・ホスティングサービス

背景(1)

- キャッシュサーバの限界
  - キャッシュが効くのは，本当に人気のあるコンテンツのみ
  - 個々のコンテンツプロバイダの要求には答えられない
    - 特定のコンテンツだけの高速化は不可能
- ミラーサーバは高コスト
  - 複数個所へのホスティング
    - 全コンテンツを持たせる必要性
  - 代表サーバにしかアクセスは集まらない
    - 複数のURLは持ちたくない
- バックボーンと足回り間の中継線の値段は下がらない
  - WDMの恩恵を受けられない

背景(2)

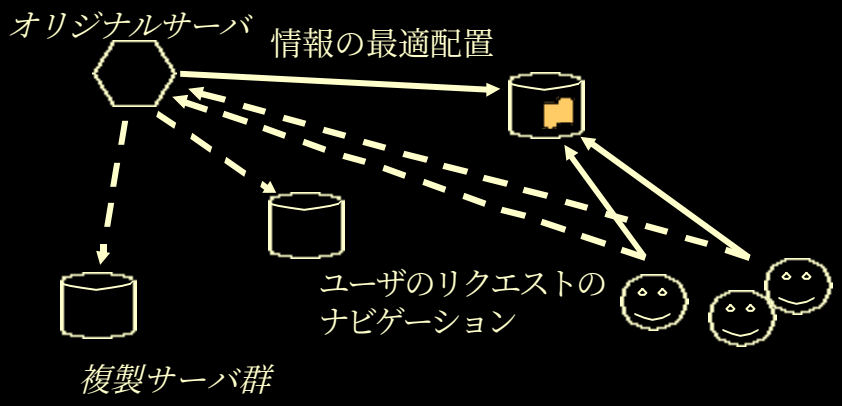
- ストリームの配信には，まだまだ帯域が不足
  - ストリームのトラフィックはこれから
    - 品質不足
      - 一般の人には耐え難い品質
      - 帯域不足で品質（コンテンツサイズ）を上げられない
    - いつか爆発(コンテンツサイズ，リクエスト数)
  - Windows Media Broadband Jumpstart initiative
    - 広帯域通信の推進
    - 主要プレイヤー
      - Content-delivery networks, Caching systems
      - DSL access, Cable access
      - Content developers and distributors, Internet advertising services

概要(1)

- 情報の最適配置
  - 人気コンテンツを，そのコンテンツを必要としているユーザの近くに配置
  - 情報の更新の検出
    - コンテンツは通常のファイルシステム上にある
- ユーザリクエストのナビゲーション
  - ユーザリクエストを最適な複製サーバに送る
    - ネットワーク距離，混雑度
    - サーバ負荷

概要(2)

- 概要



情報の最適配置

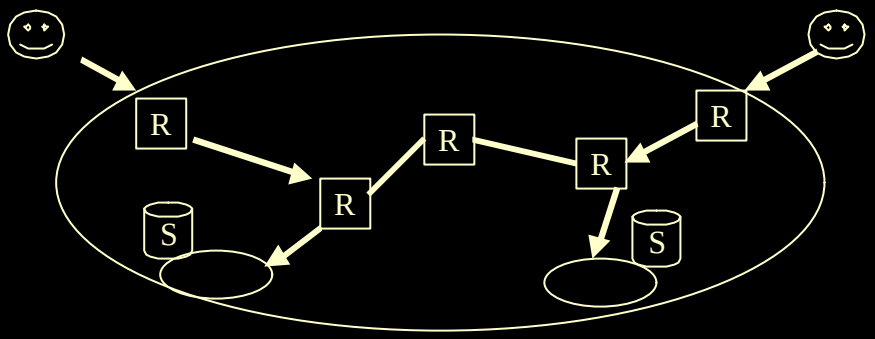
- 全ミラー
- 部分的な複製
  - パッシブ
    - リバースプロキシ
      - 人気コンテンツがキャッシュ上に残る
      - 古い情報が残る可能性
  - アクティブ
    - 計画的な配置
    - 更新情報の配信
      - 古い情報を提供する可能性をなくす

ユーザリクエストのナビゲーション

- アプリケーション自体によるリクエストのリディレクト
  - 各サービス会社独自サーバー
- DNSによる振分け
- L4スイッチ
  - 広域におけるリディレクトも可能に
- Anycast
  - ATM ?
  - IPv6
  - GlobalFeed

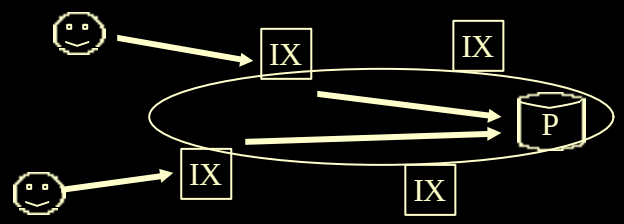
GlobalFeed

- ルーティングによるリクエストのナビゲーション
  - 同一AS内の複数の同一アドレスのサブネット
    - AS外からのトラフィックは近いサブネットに経路制御される



その他のトレンド (1)

- 高速な専用バックボーン
  - 主要IXに接続された世界規模の専用バックボーン
    - サービスを受けるサーバへのトラフィックだけが流れる
  - 遅い一般のバックボーンを避ける

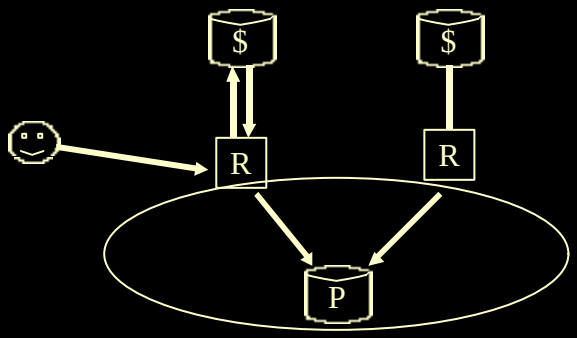


その他のトレンド (2)

- 衛星を使ったコンテンツの配信
  - まだ衛星の方が安い
  - 人気コンテンツをISPのキャッシュに押し込む
    - ブースター的

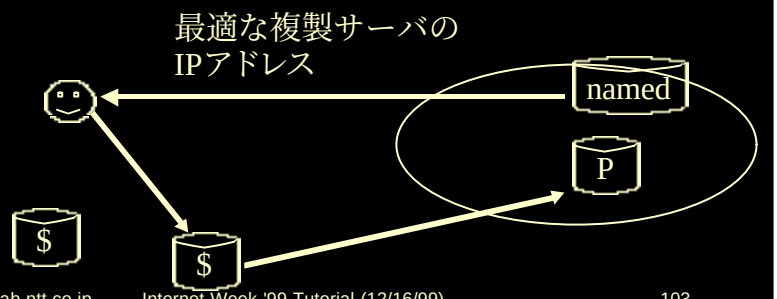
アーキテクチャ(1)

- 専用バックボーン＋透過型キャッシュ
  - 各接続ポイント（エッジ）に透過型キャッシュ



アーキテクチャ(2)

- DNS＋リバースプロキシ
  - Resolve時に最適な複製サーバのアドレスを返す
  - 複製サーバは，リバースプロキシとしてリクエストを処理する



アーキテクチャ(3)

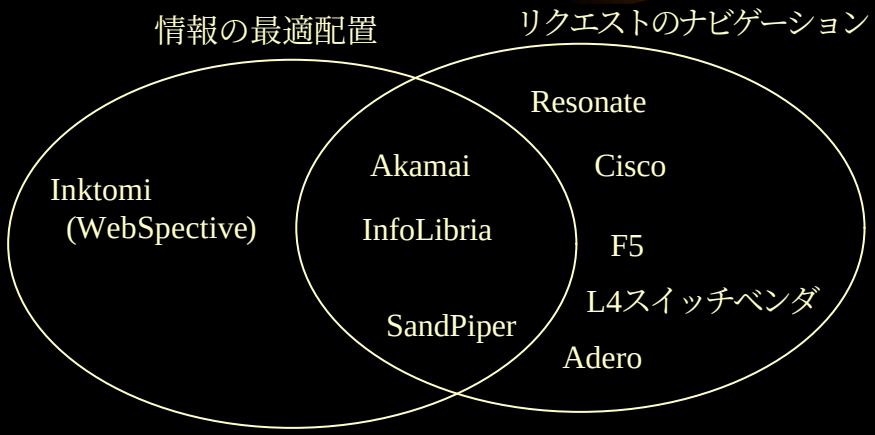
- リディレクション＋リバースプロキシ
  - アプリケーションによるリクエストのリディレクション
    - HTTP: moved temporally命令
    - Stream: サーバ指定
  - 複製サーバは，リバースプロキシとしてリクエストを処理する

ベンダ

- Akamai（サービスも行う）
  - WWW指向
  - DNSによるリクエストのナビゲーション
  - コンテンツの最適配置
- Inktomi
  - リクエストのナビゲーションには他の技術が必要
  - コンテンツの最適配置
- InfoLibria
  - ストリーム指向
  - ストリームサーバによるリクエストのナビゲーション
  - コンテンツの最適配置

ベンダ

- 分類



サービス会社(Digital Island/SandPiper)

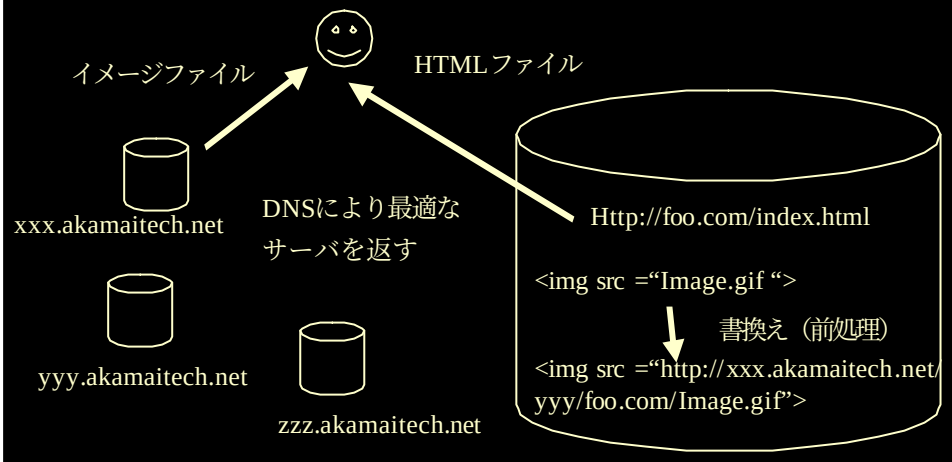
- Digital Island
  - 専用バックボーン+データセンター（５ヶ所）
  - e-Business指向
- SandPiper社 FootPrint サービス
  - サーバ数：１２００
  - Digital Islandに買収される
  - 顧客
    - E*Trade, Cisco Systems, Intuit, Novell, CNBC.com, Mastercard, Value America, Microsoft 等

サービス会社(Akamai)

- Akamai 社
  - FreeFlowサービス：イメージファイルのみ
    - イメージファイルをFreeFlowサーバ上に置く
    - DNS Resolveで最適なサーバを返す
      - Ciscoも出資済み

サービス会社(Akamai)

- システム概要



サービス会社(Akamai)

- FreeFlowサーバ
  - 1200サーバ，50サイト
    - 日本ではJPIXに2台
      - OCN等にもコンテンツをフィードしている
  - Linuxベースの独自システム
  - Cache Interface Protocol
    - 主要キャッシュベンダ製品も使用可能
  - 運用コストはISPが持つ
  - 顧客
    - CNN, Yahoo, Apple, Go Network 等

サービス会社(Akamai)

- ビジネスモデル
  - ホスティング費用をISPに持たせている
    - 回線コスト削減の見返り
    - いつまでISPにホスティング費用を持たせられるか
      - 現状
        - WWWコンテンツのみ
        - サービスしているコンテンツも少ない
      - 将来
        - ストリーミング（高価なサーバが必要）
      - ストリーミングの配送が本格的に始まると，自持ちのサーバを主要ポイントにホスティングする可能性あり

サービス会社(衛星系)

- 衛星系
  - SkyCache社
    - Akamaiと提携
    - 衛星系として老舗
  - Edgix社，HotMediaサービス
    - Dell Internet Caching Applianceサーバを対象
  - iBeam社
    - ストーリム指向

サービス会社(その他1)

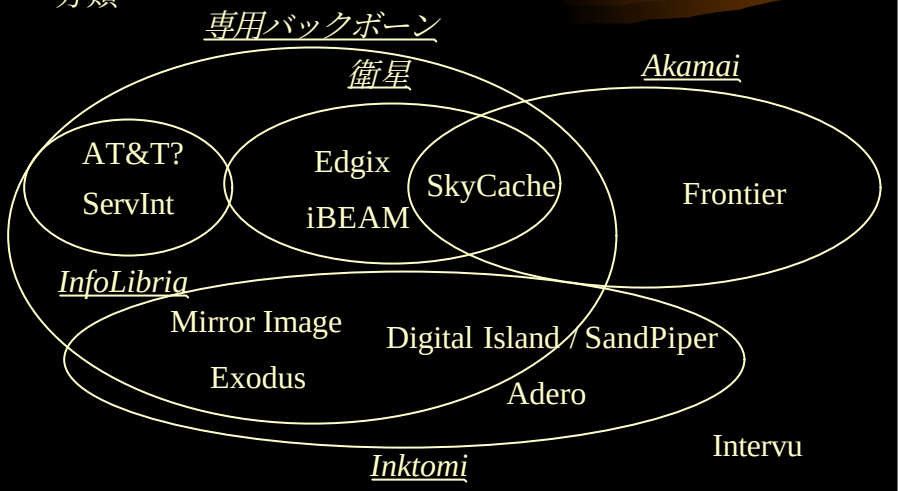
- Frontier Global Center社
  - FreeFlowサービスのリセラー
  - Global Crossing社（海底ケーブル屋）に買収される
- Adero社，AderoWorldサービス
  - DNSによるナビゲーション
- MirrorImage社，instaDeliveryサービス
  - 元商用キャッシュサービス会社
  - 値段で勝負？（$1,000 per 1M bps）

サービス会社(その他2)

- Exodus社 ReadyCache サービス
  - 元ホスティングサービス屋
  - 透過型キャッシュによるナビゲーション
  - Service Metrics Inc(パフォーマンス・モニタリング会社）を買収
- Intervu
  - ストリーミング技術ベンダ
  - 自前のネットワーク
- ServInt
  - バックボーン，ホスティング屋
- AT&T
  - ストリーム指向

サービス会社

- 分類



次は？

- ディストリビューションサービス
  - 静的なコンテンツの配信

- リアルタイムストリーム
  - 最適な配送方法
- トランザクション
  - 上手く分散できる？？
- 放送系
  - 既存の放送インフラを置換える？

まとめ

- 正しく使えば有効な技術
  - しかし，過度の期待は裏切られる
- キャッシュ（複製）技術は終わらない
  - キャッシュサーバ -> ディストリビューションサービス
  - WWW -> Stream

参考文献

- 参考文献リンク
  - http://cache.jp.apan.net/people/nabe/pr/IW99/ref.html