ITフォーラム 2014
気象庁XMLを蓄積してみて直面した課題の報告

2014/02/04

クラウド・テクノロジー活用部会 リーダー
荒本道隆
（アドソル日進株式会社）

気象庁XMLとは

- 気象庁は、これまで、**気象警報、津波警報、地震情報等、それぞれの防災情報**毎に情報の性質・利用形態などを考慮し、それぞれの情報で個別の、気象庁独自の 電文形式（フォーマット）を作成してきました。この方式は、防災情報の種類が少なく、情報の伝達がFAXや低速の通信回線の時代はそれぞれの情報に適して いましたが、高度にICT化された現在社会において、より詳細で高度化された防災情報をより効果的に活用していただくために、新たな防災情報の提供様式を 検討すべきと考え、「気象庁防災情報**XMLフォーマット**」を策定することとし、平成23年5月12日より運用を開始しました。

http://xml.kishou.go.jp/

気象庁XMLの配信

- 試行的だが、無料で配信してもらえる
  - 個人でもユーザー登録できる
  - 試行なので、遅延もありえる
- GoogleのPubSubHubbub経由で配信される
  - feed形式で、概要と気象庁XMLのURLを配信
    - そのURLにアクセスして、気象庁XML本体を取得
    - 1つのfeedの中に、複数のentryが入っている
- 注意事項
  - 受信にはSubscriberを立てて、気象庁にURLを提出
    - サンプルはGoogleCodeにある
  - 気象庁XMLは24時間以内に削除される
    - 保存しておきたければ、取り込む必要がある
    - URLはハッシュ値らしいので、重複もありえる

feedの例

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xml:lang="ja">
<title>JMAXML publishing feed</title>
<subtitle>this feed is published by JMA</subtitle>
<updated>2013-01-01T00:03:01+09:00</updated>
<id>urn:uuid:d38e0e80-12ba-3236-b10f-256b78a08995</id>
<link href="http://www.jma.go.jp/" rel="related"/>
<link href="http://xml.kishou.go.jp/feed/other.xml" rel="self"/>
<rights>Published by Japan Meteorological Agency</rights>

<entry>
<title>特殊気象報</title>
<id>urn:uuid:47516421-0ea7-331d-a1ff-1e8bf0605992</id>
<updated>2012-12-31T15:02:00Z</updated>
<author><name>室蘭地方気象台</name></author>
<link href="http://xml.kishou.go.jp/data/47516421-0ea7-331d-a1ff-1e8bf0605992.xml" type="application/xml"/>
<content type="text">【特殊気象報（風）】</content>
</entry>
<entry>
<title>特殊気象報</title>
<id>urn:uuid:d70c1366-5add-37d2-8635-ca244bea6ce3</id>
<updated>2012-12-31T15:02:00Z</updated>
<author><name>旭川地方気象台</name></author>
<link href="http://xml.kishou.go.jp/data/d70c1366-5add-37d2-8635-ca244bea6ce3.xml" type="application/xml"/>
<content type="text">【特殊気象報（風）】</content>
</entry>
</feed>
```

気象庁XMLの例

```
<?xml version="1.0" encoding="UTF-8"?>
<Report xmlns="http://xml.kishou.go.jp/jmaxml1/" xmlns:jmx="http://xml.kishou.go.jp/jmaxml1/">
  <Control>
    <Title>特殊気象報</Title>
    <DateTime>2012-12-31T15:02:00Z</DateTime>
    <Status>通常</Status>
    <EditorialOffice>室蘭地方気象台</EditorialOffice>
    <PublishingOffice>室蘭地方気象台</PublishingOffice>
  </Control>
  <Head xmlns="http://xml.kishou.go.jp/jmaxml1/informationBasis1/">
    <Title>特殊気象報（風）</Title>
    <ReportDateTime>2013-01-01T00:02:00+09:00</ReportDateTime>
    <TargetDateTime>2012-12-31T23:40:00+09:00</TargetDateTime>
    <EventID>20130101000200_風_47426</EventID>
    <InfoType>発表</InfoType>
    <Serial />
    <InfoKind>特殊気象報</InfoKind>
    <InfoKindVersion>1.0_0</InfoKindVersion>
    <Headline>
      <Text />
    </Headline>
  </Head>
  <Body xmlns="http://xml.kishou.go.jp/jmaxml1/body/meteorology1/" xmlns:jmx_eb="http://xml.kishou.go.jp/jmax
ml1/elementBasis1/">
    <MeteorologicalInfos type="特殊気象報（風）">
      <MeteorologicalInfo>
        <DateTime significant="yyyy-mm-ddThh:mm">2012-12-31T23:40:00+09:00</DateTime>
```

配信の種類と数（2013/01/01～2014/01/24）

″府県天気概況″	85067
″府県天気予報″	76693
″気象警報・注意報″	48803
″府県週間天気予報″	43326
″気象特別警報・警報・注意報″	18813
″地方海上警報″	12317
″地方海上予報″	9355
″地方週間天気予報″	8606
″府県気象情報″	7634
″紫外線観測データ″	6583
″震源・震度に関する情報″	2474
″特殊気象報″	2173
″噴火に関する火山観測報″	2066
″地方気象情報″	1990
″台風解析・予報情報（３日予報）″	1718
″全般海上警報（定時）″	1561
″生物季節観測″	1535
″竜巻注意情報″	1500
″府県高温注意情報″	1412
″土砂災害警戒情報″	992
″指定河川洪水予報″	830
″地方１か月予報″	627
″全般台風情報（定型）″	597
″全般週間天気予報″	458
″異常天候早期警戒情報″	354
″震度速報″	352
″台風解析・予報情報（５日予報）″	301
″全般海上警報（臨時）″	287
″府県海氷予報″	268
″地方高温注意情報″	259
″震源に関する情報″	250
″府県潮位情報″	223
″火山の状況に関する解説情報″	215
″季節観測″	210
″全般気象情報″	200
″スモッグ気象情報″	193
″府県天候情報″	186
″地方３か月予報″	132
″全般台風情報″	122
″全般台風情報（詳細）″	98
″記録的短時間大雨情報″	77
″地方天候情報″	71
″地方潮位情報″	65
″全般１か月予報″	57
″全般潮位情報″	28
″地方暖・寒候期予報″	22
″津波情報″	15
″全般天候情報″	14
″東海地震観測情報″	13
″全般３か月予報″	12
″地震の活動状況等に関する情報″	10
″津波警報・注意報・予報″	9
″噴火警報・予報″	8
″気象特別警報報知″	7
″津波警報・注意報・予報a″	7
″津波情報a″	6
″全般スモッグ気象情報″	5
″顕著な地震の震源要素更新のお知らせ″	5
″火山現象に関する海上警報・海上予報″	3
″全般暖・寒候期予報″	2

SPARQLの画面から参照可能

蓄積した理由

- とりあえず貯めてみよう
  - 毎日、無料で本物データが配信される
    - 本物, オープン, かなりの量,
  - 後から入手できないかもしれない
    - 気象データは、いつかどこかで必ず使うはず
- クラウド上の仮想サーバを利用
  - グローバルIPを持った受信サーバが必要

蓄積方法

- ファイルシステム
- メッセージキュー
- RDB/RDF

Step1：ファイルシステムに蓄積

- 受信サーバ内のHDDにファイルで保存
  - 日付でディレクトリを作成
- 集計期間：2013/01/01 ～ 2013/12/31
  - feed：受信年月日時分秒をファイル名に保存
    - 572,595件／年（平均1,568件／日）
    - 2,444MByte／年（平均6.69MByte／日）
  - 気象庁XML：URLのファイル名のまま保存
    - **323,212件／年**（平均885件／日）
    - **6,399MByte／年**（平均17.53MByte／日）
- 問題発生
  - 内容のまったく同じfeedが何度も飛んでくる

Step2：メッセージキュー

- メッセージキューを利用して配信
  - Kafkaに蓄積
    - 1対nでキュー管理ができるメッセージングシステム
    - 古いものから自動的に消えていく
  - WebSocket/Cometで配信
    - ほぼリアルタイムに社内に配信してみよう

WebSocketとは

- **WebSocket**（ウェブソケット）は、コンピュータ・ネットワーク用の通信規格の1つである。インターネットの標準化団体であるW3CとIETFがウェブサーバーとウェブブラウザとの間の通信のために規定を予定している双方向通信用の技術規格であり、APIはW3Cが、WebSocket プロトコルはIETFが策定に関与している。プロトコルの仕様は RFC 6455。TCP上で動く。

Wikipediaより

- Comet（Ajax）の代替として、策定された

- **Comet**（コメット）とは、Web アプリケーションを構築する際に利用される技術で、この技術を使うと、サーバで発生したイベントをクライアントからの要請なしにクライアントに送信することができる。
- Comet はこのような通信を実現するための複数の手法をまとめた概念である。これらの手法はブラウザにプラグインを追加することなく、（JavaScript のような）デフォルトの機能で実現されるものである。理論的には Comet は、ブラウザがデータを要求する形の既存のウェブのモデルとは異なっている。実際は Comet アプリケーションは Ajax と Long polling を使用してサーバ上の新規データを取得する。

Wikipediaより

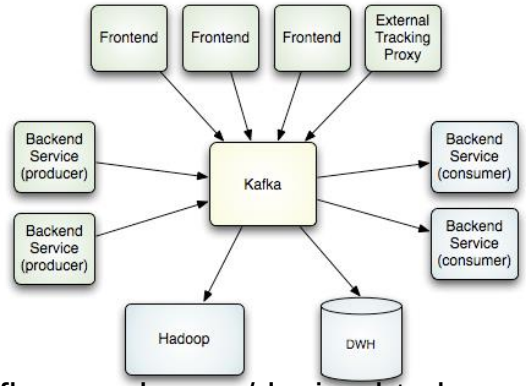
- WebSocketの状況
  - HTML5から切り離された
  - HTTP2.0は、対抗規格のSPDYが優位

実装方法

- サーバにはAmazonEC2を使用
- Subscriber
  - GoogleCodeからJavaのものを使用
    - http://code.google.com/p/pubsubhubbub-java/
  - Mainクラスを追加
  - PuSHhandler.javaに追記
    - if (request.getMethod().equals("POST")) {

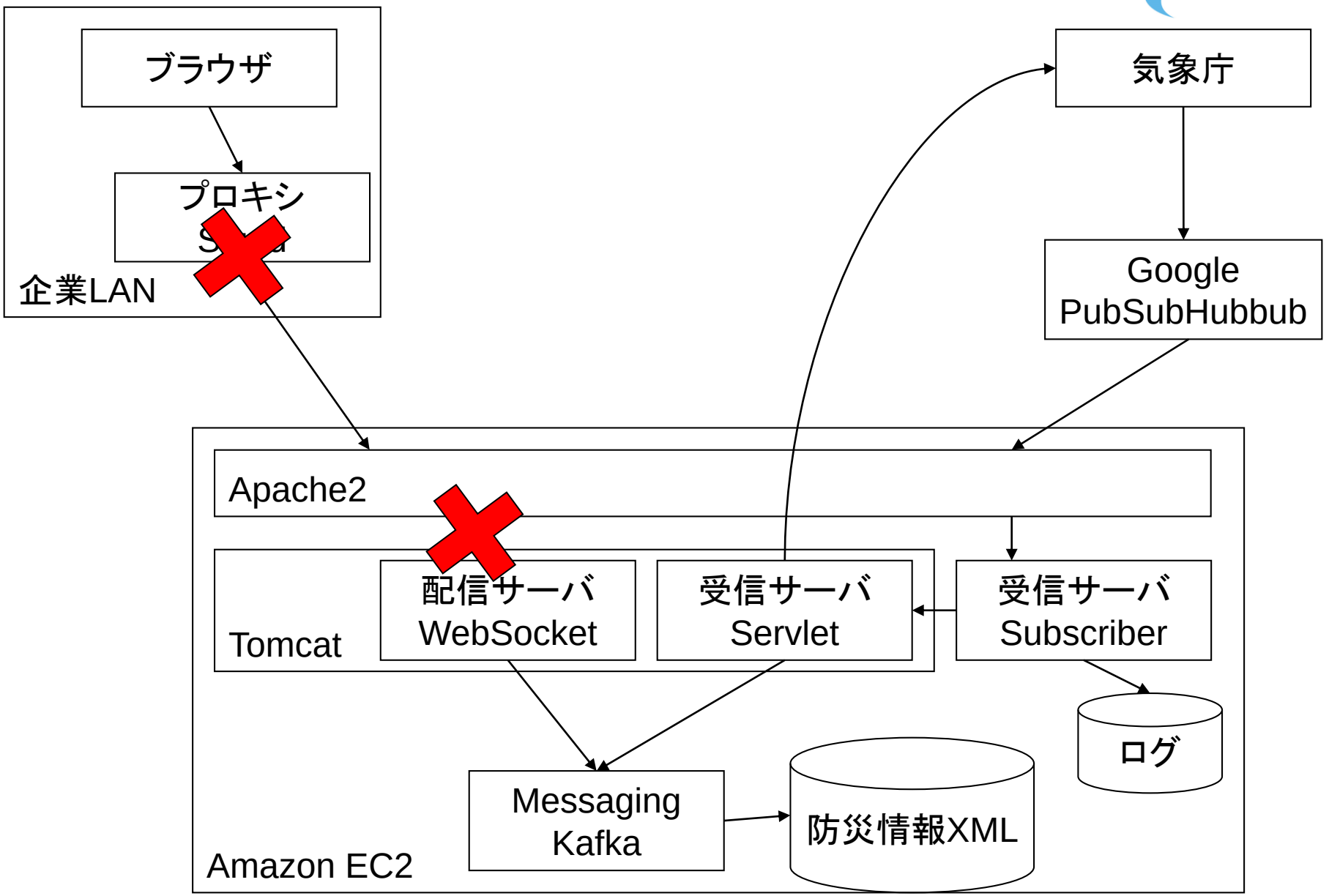
```
public static void main(String[] args) {
  PubSubHubbub.Web web
     = new PubSubHubbub.Web(8888);
  web.start();
}
```

- WebSocket
  - Tomcat7のWebSocketServletを使用
    - org.apache.catalina.websocket.WebSocketServlet
- Kafka（messaging system）
  - SimpleConsumerを使用
    - 1つのメッセージを複数から取り出せる
      - Topic単位で古いものが消えていく



http://kafka.apache.org/design.html

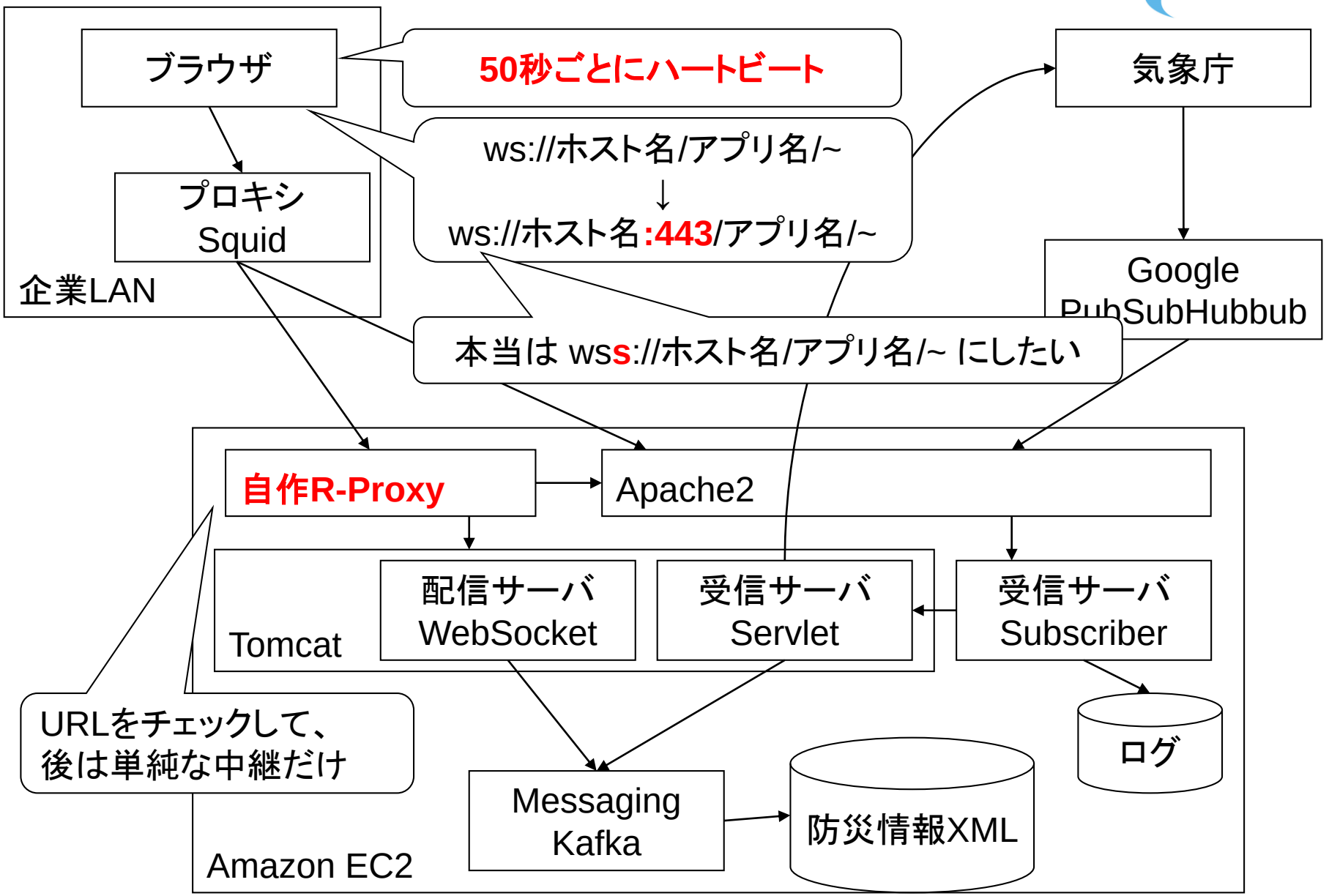
配信システムの構成



WebSocketに関連する問題が次々と発生

- Squidがあると
  - ブラウザはCONNECTを使う
    - CONNECT cloud.projectla.jp:80 HTTP/1.1
    - Squidのデフォルト設定：**CONNECTは443以外を禁止**
- Apache2を入れると
  - セキュリティ確保や微調整をApache2でやりたい
  - 接続したままを想定していない
    - 「HTTP/1.1 101 Switching Protocols」で**子httpdが落ちる**
- Amazon EC2のロードバランサを入れると
  - **60秒でタイムアウト**するらしい

WebSocketの問題を解決



Kafkaでも問題発生

- 新規クライアントからのアクセス時
  - 全メッセージの先頭から順次空読み
  - offset値の最新50件だけをリストに残す
  - 最後まで読んだら、リストにある50件を返す
  - 以降、メッセージが増えれば、すぐにそれを返す
- Kafkaの特徴
  - 一定量のメッセージが溜まると、古い物から消す
- ２週間ほど連続運転してみた結果
  - 先頭（offset=0）が消去された
  - 先頭が無いので、**先頭から空読みができない**
    - メモリ上でキャッシュしていたので、Tomcat再起動時に発生
  - APIを調べても、「最後から読む」が無い？
  - 解決方法
    - 「最新から50番目のoffset値」を常にファイルへ保存
      - 空読みが不要になり、性能も改善

WebSocketとCometの比較

- 配信時の通信量
  - WebSocket：0Byte送信、3Byte+XML受信
  - Comet：400Byte送信、284Byte+XML受信
- ハートビート時（50秒間隔）の通信量
  - WebSocket：6Byte送信、0Byte受信
  - Comet：400Byte送信、245Byte受信

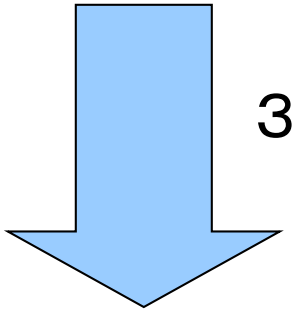
- 24時間の合計送受信サイズ
  - WebSocket：10KByte送信、2.6KByte+XML(17,530KByte)受信
  - Comet：1,045KByte送信、674KByte+XML(17,530KByte)受信

Step3：RDB/RDFに蓄積

- RDB
  - PostgreSQL Server 9.3.1
  - 全feedを保存し、検索用REST APIを提供
  - Xpathでインデックスを作成
- RDF
  - Apache Jena 2.11
  - 全feedをRDFに保存し、SPARQLで集計
  - アプリが集計情報を表示

データが溜まってくると問題が発生

- RDB
  - インサート時に、長時間のHDDアクセスが発生
    - Xpathによるインデックス作成が原因？
- RDF
  - 集計にかかる時間が長く、アプリの応答が遅くなる
    - 新データが届くたびに事前集計し、キャッシュに保存



3ヵ月後

- feedが届くたびに、数十秒間HDDをアクセス
  - 1,568回／日 → 1,568 x 20秒 ＝ 約8.7時間／日

その結果....

- 某A社の仮想サーバを利用
  - インスタンス代：約$27/月
- I/O課金が止まらない

2013/06	$0.55	システム構築
2013/07	$1.51	運用開始
2013/08	$25.22	RDFの性能低下により、事前集計を導入
2013/09	**$214.12**	RDF+RDBによるHDDアクセス
2013/10	$158.38	請求額に驚いて、RDFの事前集計を外した
2013/11	$68.76	RDBによるHDDアクセス
2013/12	$5.70	固定料金のサービスに引越し完了

- 解決方法
  - ストレージの種類を変える方法もあったが
  - 完全固定料金の仮想サーバに引っ越した

まとめ

- 運用システムと接続すると、色々な事件が起こる
  - 壊れたfeedが飛んでくる
  - 同じfeedが何度も飛んでくる
  - 想定していなかったデータ構造
- データがある程度溜まると、色々な事件が起こる
  - 性能劣化
  - 従量課金
    - I/O課金の根本的な原因は、システム構成の設計
    - ネットワーク課金は心配するほど無かった
- せっかく気象庁がデータを配信してくれているので
  - 受信しなければ勿体ない
  - 数十万件のデータを扱う練習にもなる
- 今後、バイナリデータも扱うとなると
  - テキストデータとは違ったノウハウが必要になる
    - 気象庁で使っているGRIB2は、圧縮されている

現在の状況

- Yahoo Open Hack Day Japan 2 にAPIを提供
  - 2014/02/15～16

http://cloud.projectla.jp/ohd2/