インターネット計測とデータ解析 第3回

長 健二朗

2011年5月25日

前回のおさらい

インターネットのサイズを計る

- ユーザ数、ホスト数
- ウェブページ数
- 精度 誤差 有効数字
- グラフによる可視化
- 演習:gnuplot によるグラフ描画

今日のテーマ

データの記録とログ解析

- データフォーマット
- ログ解析手法
- 演習:ログデータと正規表現

いろいろなログ

- web server accesslog
- mail log
- syslog
- firewall log
- IDS log
- その他 あらゆる記録

なぜログ解析をするのか？

- 現状の把握
  - 新しい発見: 技術の進歩や利用形態の変化
  - そのうえで将来予測
- セキュリティ上の問題や機器故障、それらの兆候の把握
- 解析技術の向上
  - 自動化
- 障害のレポート、問題への対応
- 記録の必要性
  - 法的理由、その他

そもそも解析されないログには意味がない
(ログを取るだけで安心してはいけない)

ログ解析の問題

- 膨大なデータ量
- 必要な情報や精度の欠如、時刻情報や内容の信憑性
- (収集システムの障害などによる) 記録の欠落
- さまざまなフォーマットが存在
- 解析には時間と労力が必要
- 解析は難しいという思い込み

ログの管理

- ログ収集
  - プログラミング (syslog API の利用など)
  - 収集システム構築
- ログローテーション
  - 古いデータを一定期間保存した後削除
  - ログサイズ、時間、データの古さ
  - ローテーション時にデータを失わないよう工夫
- RRD (Round Robin Database)
  - 古いログを集約することで、データサイズを一定にする
  - 例: 5 分粒度で 1 週間、2 時間粒度で 1 カ月、1 日粒度で 1 年
- 可視化
  - グラフ化して web に貼ることで状況の把握を容易に

さまざまなフォーマット

- web server access log
- mail log
- DHCP server log
- syslog

web server access log

- Apache Common Log Format
  - client_IP client_ID user_ID time request status_code size
- Apache Combined Log Format
  - Common Log Format に referer と User-agent を追加
  - client_IP client_ID user_ID time request status_code size referer user-agent
- その他 カスタマイズ可能

client_IP: アクセス元の IP アドレス
client_ID: クライアントの識別子
user_ID: 認証ユーザ名
time: 時刻
request: リクエストの最初の行
status_code: レスポンスステータス
size: 送信バイト数 (ヘッダーは含まず) 0 バイトだと"-"
referer: リクエストのリンク元
user-agent: リクエスト元のブラウザの種類やバージョン

例 Combined Log Format:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] \
  "GET /apache_pb.gif HTTP/1.0" 200 2326 \
  "http://www.example.com/start.html" \
  "Mozilla/4.08 [en] (Win98; I ;Nav)"
```

mail log

受信、送信などのメール処理毎にログが取られる
例:

```
Oct 27 13:32:54 server3 sm-mta[24510]: m9R4WsBe024510:\
 from=<client@example.com>, size=2403, class=0, nrcpts=1 \
 msgid=<201012121547.oBCFlPX6032787@example.com>, \
 proto=ESMTP, daemon=MTA, relay=mail.example.co.jp [192.0.2.1] \
Oct 27 14:43:04 server3 sm-mta[24511]: m9R4WsBe024510: \
 to=<user@example.co.jp>, delay=01:10:10 xdelay=00:00:00, \
 mailer=local, pri=32599, dsn=2.0.0, stat=Sent
```

- 時刻
- ホスト名
- プロセスオーナー [プロセス番号]
- Queue ID: メールの内部 ID
- ...
- nrcpts: 受信者数
- relay: 次の送信先サーバ
- dsn: Delivery Status Notification, RFC3463
  - 2.X.X:Success, 4.X.X:Persistent Transient Failure, 5.X.X:Permanent Failure
- stat: Message Status
  - Sent, Deferred, Bounced, etc

DHCP server log

SYSLOG: メッセージの記録

```
Oct 28 15:04:32 server33 dhcpd: DHCPDISCOVER from 00:23:df:ff:a8:a7 via eth0
Oct 28 15:04:32 server33 dhcpd: DHCPOFFER on 192.168.2.101 \
  to 00:23:df:ff:a8:a7 via eth0
Oct 28 15:04:32 server33 dhcpd: DHCPREQUEST for 192.168.2.101 \
  from 00:23:df:ff:a8:a7 via eth0
Oct 28 15:04:32 server33 dhcpd: DHCPACK on 192.168.2.101 \
  to 00:23:df:ff:a8:a7 via eth0
Oct 28 15:09:32 server33 dhcpd: DHCPREQUEST for 192.168.2.101 \
  from 00:23:df:ff:a8:a7 via eth0
Oct 28 15:09:32 server33 dhcpd: DHCPACK on 192.168.2.101 \
  to 00:23:df:ff:a8:a7 via eth0
```

dhcpd.leases: 割り当てた IP アドレスの個別情報

```
lease 192.168.100.161 {
  starts 4 2010/12/09 23:13:39;
  ends 5 2010/12/10 00:13:39;
  tstp 5 2010/12/10 00:13:39;
  binding state free;
  hardware ethernet 5c:26:0a:17:06:00;
}
```

syslog

- UNIX 系 OS で任意のメッセージを通知したり保存する仕組み
  - もともとメールの処理記録保存用だったが広く使われるようになった
  - 他のサーバに転送も可能
  - ログのローテーション機能のサポート
- Windows Event Log

ログ解析手法

- 思いつくことを色々試す、グラフ化する
  - 手を動かしている内に分かる事、思いつく事が多い
- 処理スクリプトとコマンドラインツール (grep, sort, uniq, sed, awk, etc)
- 大量データを効率よく処理する工夫
- 繰り返し行う処理はできるだけ自動化する
  - いっぽうで自動化した処理を過信しないこと

大量データの扱い

- ナイーブにやると膨大なデータの読み込みや処理テーブルが必要
  - データ構造やアルゴリズムを勉強しておくと役立つ
- 大量データを扱う工夫
  - 集計に不要な情報の削除
  - 時間的、空間的に集約
  - 必要に応じて分割処理
  - 必要に応じて分散並列処理
- 中間ファイルに変換する、分割処理する
- 処理に必要なメモリ量の見積り
  - データ構造を工夫する
  - 一度に処理するサイズ、次元を押える工夫
- 全体の処理時間の見積り
  - 小さいデータセットで試行
  - スケールするアルゴリズム
- メモリサイズと処理時間のトレードオフに配慮

基本的手法

- 正規表現
- 集約
- 正規化
- 閾値
- 外れ値の扱い

Web access log サンプルデータ

- SFC ITC が管理する SFC 公式 web サーバへのアクセスログ
- SFC-SFS の授業ページから配布
  - weblog-20110228-20110306.txt.zip (4.6MB 展開時 48MB)

- データの取り扱いに関する注意
  - 本科目の演習以外の用途に使用しない
  - 本科目履修者以外への配布禁止
  - 科目終了時にデータを破棄すること
  - データ配布対象者 (履修者) 名簿を SFC 地球広報委員会に提出済

- 個人情報保護への配慮からデータには以下の加工を行っている
  - 学生および教員の個人用ページは対象から除外
  - IP アドレスは匿名化 (1 対 1 マッピング、prefix 対応保存)
  - apache common log format ("referer", "user-agent" はなし)
  - remote login name (%l), remote user (%u) は削除
  - 検索キーワードの削除 ("search.html?" に続くキーワード)
  - 一部の誤入力が疑われるエントリーを削除 ("˜user" など)

サンプルデータ内容

```
52.99.79.208 - - [28/Feb/2011:00:00:04 +0900] "GET /top.html HTTP/1.1" 200 9947
178.194.177.22 - - [28/Feb/2011:00:00:05 +0900] "GET / HTTP/1.1" 304 -
178.194.177.22 - - [28/Feb/2011:00:00:05 +0900] "GET /images/pen.gif HTTP/1.1" 304 -
178.194.177.22 - - [28/Feb/2011:00:00:05 +0900] "GET /images/keiou.gif HTTP/1.1" 304 -
178.194.177.22 - - [28/Feb/2011:00:00:05 +0900] "GET /images/copy.gif HTTP/1.1" 304 -
178.194.177.22 - - [28/Feb/2011:00:00:07 +0900] "GET /flash/ HTTP/1.1" 200 3269
174.177.94.5 - - [28/Feb/2011:00:00:08 +0900] "OPTIONS * HTTP/1.0" 200 -
52.99.79.208 - - [28/Feb/2011:00:00:08 +0900] "GET /alumni/ HTTP/1.1" 200 6366
178.194.177.22 - - [28/Feb/2011:00:00:09 +0900] "GET /top.html HTTP/1.1" 200 9947
52.99.79.208 - - [28/Feb/2011:00:00:09 +0900] "GET /students_soukan/ HTTP/1.1" 200 9836
233.41.145.151 - - [28/Feb/2011:00:00:12 +0900] "GET /academics/undergraduate/ei/faculty.html HTTP/1.1" 20
67.127.37.169 - - [28/Feb/2011:00:00:14 +0900] "GET /en/aboutsite.html HTTP/1.0" 200 7606
52.99.79.208 - - [28/Feb/2011:00:00:14 +0900] "GET /alumni/certificates.html HTTP/1.1" 200 5993
98.37.193.251 - - [28/Feb/2011:00:00:14 +0900] "GET / HTTP/1.0" 200 2048
100.104.153.171 - - [28/Feb/2011:00:00:15 +0900] "GET /academics/graduate/ HTTP/1.1" 200 9274
66.187.123.187 - - [28/Feb/2011:00:00:15 +0900] "GET /academics/graduate/dd.html HTTP/1.1" 304 -
100.104.153.171 - - [28/Feb/2011:00:00:15 +0900] "GET /css/main.css HTTP/1.1" 200 11373
99.67.3.251 - - [28/Feb/2011:00:00:16 +0900] "GET /about_sfc/facts/index.html HTTP/1.1" 200 5834
174.177.94.5 - - [28/Feb/2011:00:00:17 +0900] "OPTIONS * HTTP/1.0" 200 -
67.127.37.169 - - [28/Feb/2011:00:00:17 +0900] "GET /news/20110112.html HTTP/1.0" 200 7985
...
```

正規表現 (regular expression)

正規表現

- 文字列パターンの表記法、文字列の検索や置換に利用
- もともとは形式言語理論において正規言語を表すための手段
- 文字列のパターンマッチのための記法として普及
  - grep, expr, awk, vi, lex, perl, ruby, ...

Ruby の正規表現

```
Regexp class
regular expression literal: /regexp/opt
=~ operator: subject =~ /regexp/
match() method:  /regexp/.match(subject)
string class: string.match(/regexp/)
```

Ruby の正規表現クイックレファレンス

```
[abc]  A single character: a, b or c
[^abc]  Any single character but a, b, or c
[a-z]  Any single character in the range a-z
[a-zA-Z]  Any single character in the range a-z or A-Z
^  Start of line
$  End of line
\A  Start of string
\z  End of string
.  Any single character
\s  Any whitespace character
\S  Any non-whitespace character
\d  Any digit
\D  Any non-digit
\w  Any word character (letter, number, underscore)
\W  Any non-word character
\b  Any word boundary character
(...)  Capture everything enclosed
(a|b)  a or b
a?  Zero or one of a
a*  Zero or more of a
a+  One or more of a
a{3}  Exactly 3 of a
a{3,}  3 or more of a
a{3,6}  Between 3 and 6 of a
```

Ruby の正規表現クイックレファレンス (つづき)

```
options:
i case insensitive
m make dot match newlines
x ignore whitespace in regex
o perform #{...} substitutions only once
```

最長マッチと最短マッチ (最短マッチの方が高速)

"*"や"+"は最長マッチ、"*?"や"+?"は最短マッチ

```
/<.*>/.match("<a><b><c>")  # => "<a><b><c>"
/<.*?>/.match("<a><b><c>") # => "<a>"
```

演習: サンプルデータから時系列データを抽出しグラフ化

- サンプルデータ: Web サーバアクセスログ 1 週間分
- 1 時間ごとのリクエスト数と転送量の抽出
- 結果をグラフ化する

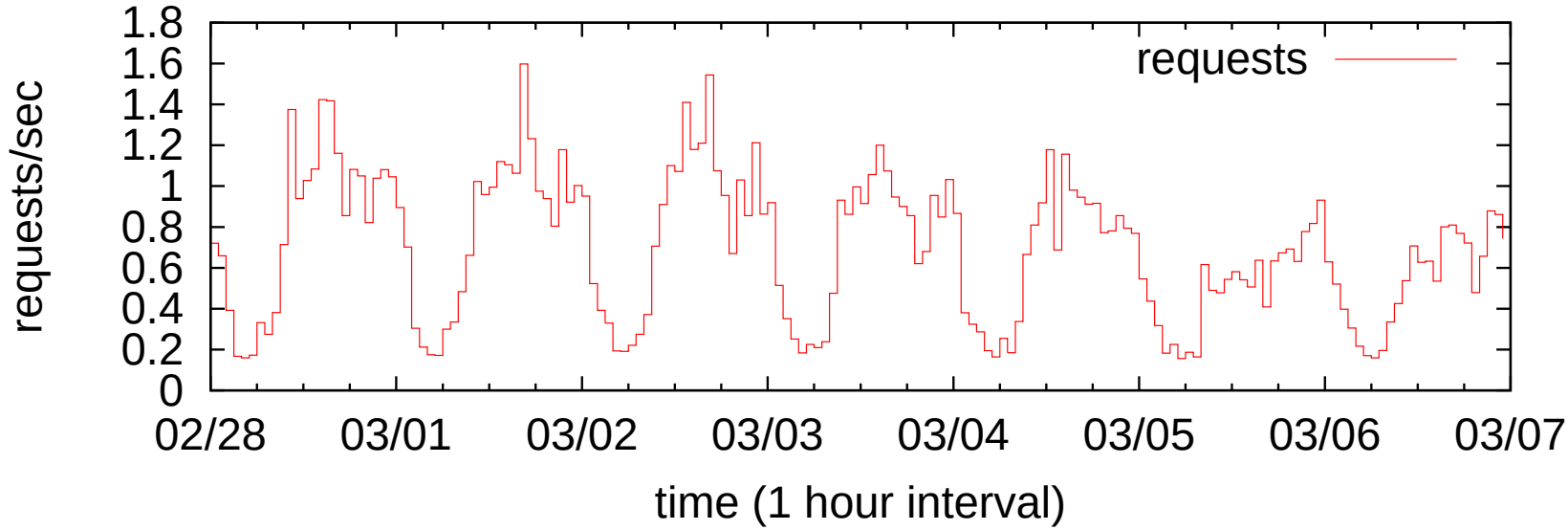
1時間ごとのリクエスト数と転送量の抽出

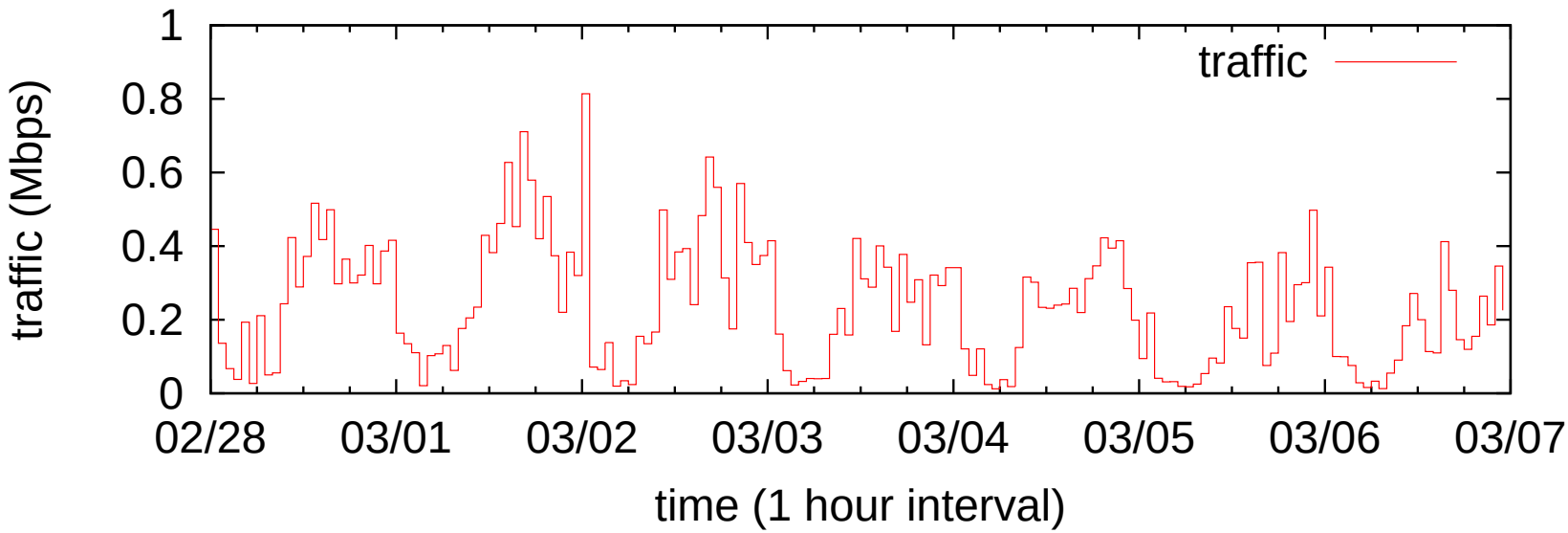
```
#!/usr/bin/env ruby
require 'date'

# regular expression for apache common log format
#   host ident user time request status bytes
re = /^(\S+) (\S+) (\S+) \[(.*?)\] "(.*?)" (\d+) (\d+|-)/

timebins = Hash.new([0, 0])
count = parsed = 0
ARGF.each_line do |line|
  count += 1
  if re.match(line)
    host, ident, user, time, request, status, bytes = $~.captures
    # ignore if the status is not success (2xx)
    next unless /2\d{2}/.match(status)
    parsed += 1
    # parse timestamp
    ts = DateTime.strptime(time, '%d/%b/%Y:%H:%M:%S %z')
    # create the corresponding key for 1-hour timebins
    key = ts.strftime("%Y-%m-%dT%H")
    # count by request and byte
    timebins[key] = [timebins[key][0] + 1, timebins[key][1] + bytes.to_i]
  else
    # match failed
    $stderr.puts("match failed at line #{count}: #{line.dump}")
  end
end
timebins.sort.each do |key, value|
  puts "#{key} #{value[0]} #{value[1]}"
end
$stderr.puts "parsed:#{parsed} ignored:#{count - parsed}"
```

1時間ごとのリクエスト数と転送量のグラフ化





gnuplot script

- multiplot を使ってプロットを並べる

```
set xlabel "time (1 hour interval)"
set xdata time
set format x "%m/%d"
set timefmt "%Y-%m-%dT%H"

set multiplot layout 2,1

set yrange [0:1.8]
set ylabel "requests/sec"
plot "time.txt" using 1:($2/3600) title 'requests' with steps

set yrange [0:1.0]
set ylabel "traffic (Mbps)"
plot "time.txt" using 1:($3*8/3600/1000000) title 'traffic' with steps

unset multiplot
```

まとめ

データの記録とログ解析

- データフォーマット
- ログ解析手法
- 演習:ログデータと正規表現

次回予定

第 4 回 インターネットの速度を計る (6/1)

- 速度計測
- 利用可能帯域の推測
- 平均 標準偏差
- 線形回帰
- 演習:平均、標準偏差、線形回帰
- **課題 1**