On collecting and using Web data

Balachander Krishnamurthy

AT&T Labs–Research, Florham Park, NJ
`http://www.research.att.com/~bala/papers`

Philosophy of using data for research

- Why not use synthetic model and generated data?
- Real data is available, software exists to process it
- Multiple sources of data (representative), now even a repository
- Applicable and applied to real problems
- No chance of deployment in IETF protocols without experiments and implementations

Overview of talk

- Web data: What is it?
- Where is it collected?
- Why is it collected?
- How is it collected?
- How is it processed?
- How have I used it?

This talk is *not* about cookies, pageviews, clickthrough etc.

What do I mean by Web data?

- Client, proxy, server logs
- Packet traces (headers, metadata)
- Raw Web pages (but not my focus)

Where is it collected?

- Everywhere
- In some sites it is being collected without *anyone* knowing about it
- Organizations of every ilk
- At Web events (Olympics, Superbowl, Mars Sojourner landing)
- Even at research sites by some researchers (like me)
- Ah yes, DIMACS as well

Why is it collected?

- Track users
- Performance evaluation of Web server, site, network
- Configuration default ("we might find a use for it")
- Variety of research uses:
  - Simulation of new algorithms
  - Traffic estimation, hot sites, hot spots inside a site
  - Improved caching, moving content closer to user

How is it collected, what format..

- Logging mechanisms built into components (clients, proxies, servers)
- External logging such as packet traces
- May need to store large amounts of data
- Elements to be logged are chosen
- Logged in a variety of formats

How is it processed?

- Often with difficulty (we will discuss problems later)
- Must retain data integrity
- Try to maintain privacy
- Reduce overhead

How have I used it?

In a variety of research projects—broadly in three categories

1. Web caching, end to end information information exchange
2. HTTP Protocol improvement
3. Web/network characterization

Details

What is Web data and where is it collected?

Where is data collected?

- At most sites—more common at server, then proxy sites
- Site survey (a la Netcraft)
- Explicit collection (a la Mediametrix) at representative client sites
- Network backbone (a la Keynote Systems) at ISPs

Location, location, location

- Diversity of data stems from point of collection
- Site survey may be more representative when followed by checks but limited nature of data gathered (server type, OS hardware platform)
- Metered homes should be representative of access speed, time of access, number of accesses etc.
- Backbone data depends on whether hosting site is close by and what percentage of overall traffic is seen there. Still may not see bi-directional traffic.

What kind of data?

- Client log: shows outgoing traffic from client site (request direction)
- Proxy log: shows shared outgoing traffic of several clients
- Server log: shows incoming traffic on a given site
- Packet traces: incoming and outgoing (at low level) but not all
- Raw Web pages—(crawled/pre-crawled date, e.g., *archive.org*)

Client logs

- Gathered at browser
- Easy to capture by turning on logging, once browser has been changed
- More useful when combined with local low level traces
- Keep track of percentage of connections aborted etc.
- Clean model of user's access pattern with no distortions (proxy hits)

Problems with client logs

- Unlike server or even proxy logs, no uniform format
- Each browser version needs to be changed or other users forced to use a modified browser
- Subset of participants may not be representative
- In many places it is easier to gather information via proxy logs
- Individual log hard to anonymize
- Relatively rare, hence not very useful except to monitor oneself

Proxy logs

- Gather accesses of many users at central entrance/exit point
- Can glean generic access pattern: best way to learn about many clients
- Very useful in identifying popular servers, peak periods of access
- Can learn about resources on wide variety of servers
- Possible to anonymize entries without losing information
- Useful in research: caching, rate of change, prefetching, resource attribute distributions (such as size, content type..)

Problems with proxy logs

- Must force clients to go through proxy (done in many places)
- Transparent proxy may bring its own set of problems
- Not all clients behind a proxy are individual users
- Accesses seen by proxy is not universe of all accesses (browser cache)
- Proxy logs reflect on local users only
- Many organizations won't willingly give out proxy logs
- Some anonymization can lead to information loss

Server logs

- Most common set of logs
- Widely gathered (and occasionally made available!)
- Requests from very wide range of clients
- Possible to characterize access popularity of resources
- Analysis can help improve server performance
- Many commercial analysis tools, some standard formats

Problems with server logs

- Difference across logs: various access patterns
- Depends heavily on content type mix at site
- Event logs vs. longer duration logs
- Cache hits (at proxy or client side) not seen in server logs
- May have problems distinguishing between spiders, proxies, clients
- Several header fields not logged, aborts not seen, timestamp coarse-grained
- Many sites will not release (even anonymized) logs

Packet traces

- Can be gathered at network access points, backbone, peering link
- Gathering transparent to other components
- Very detailed, fine-grained information compared to logs
- All HTTP, TCP level information available
- Can find out about network overheads, protocol interaction
- Complete request and response headers
- User's action like clickahead/aborts captured

Problems with packet traces

- Specialized hardware and software needed
- Monitor must be connected at interesting network location
- Link speeds are no longer FDDI/FastE but OC48/OC192
- Depending on network configuration, monitor may see only part of traffic
- May have to monitor multiple interfaces (hack *tcpdump*!)
- Volume of data is often very high, dynamic reconstruction needed to avoid swamping disk/losing packets
- Complete violation of user privacy since all bits are seen

Details

Why is it collected?

Why is it collected: details

- Tracking users, domains, countries, etc.
- Advertising and targeting purposes
- Measure of throughput
- Moving content around and revamping site
- Load balancing

Why is it collected: more details

- Traffic estimation
- Hot sites
- Better caching
- Content migration
- Content replication
- Content adaptation

Why is it really collected?

Details

How is it collected?

How is it collected: Format

- Why not have a common log format?
- Agree on what needs to be logged
- Few Web servers in practice, so few formats at most?
- Er, no.
- Easier to agree on format, but what gets logged varies

Common Log Format—CLF

- IP address or name of the client (remote host)
- Remote identity of user and authenticated username ("-")
- Date and time of the request (or service completion time)
- First line of the request including the HTTP method and URL
- HTTP response status code (200, 304, ...)
- Number of bytes in the response (Content-Length)

CLF–example

A few lines in common log format

```
122.345.131.2 - - [15/Oct/1998:00:00:25 -0400] "GET /images/wwwtlogo.gif
   HTTP/1.0" 304 -
122.345.131.2 - - [15/Oct/1998:00:00:26 -0400] "GET /images/bsite.gif
    HTTP/1.0" 304 -
13.3.2.16 - - [15/Oct/1998:00:00:38 -0400] "GET /estatements/bykmbhi.html
    HTTP/1.0" 304 -
15.16.202.34 - - [15/Oct/1998:00:00:45 -0400] "GET /images/today.gif
    HTTP/1.0" 200 3699
```

Extended Common Log Format—EChbLF

First 7 fields same as CLF.
New: `Referer` and `User Agent` fields
Plust: fields related to proxy and remote server

- Remote server's status code, response header size sent to proxy
- Proxy's response header size to client, request header size to remote server
- Client's request body and header size, forwarded request's body size
- Request processing time in seconds

ECLF–example

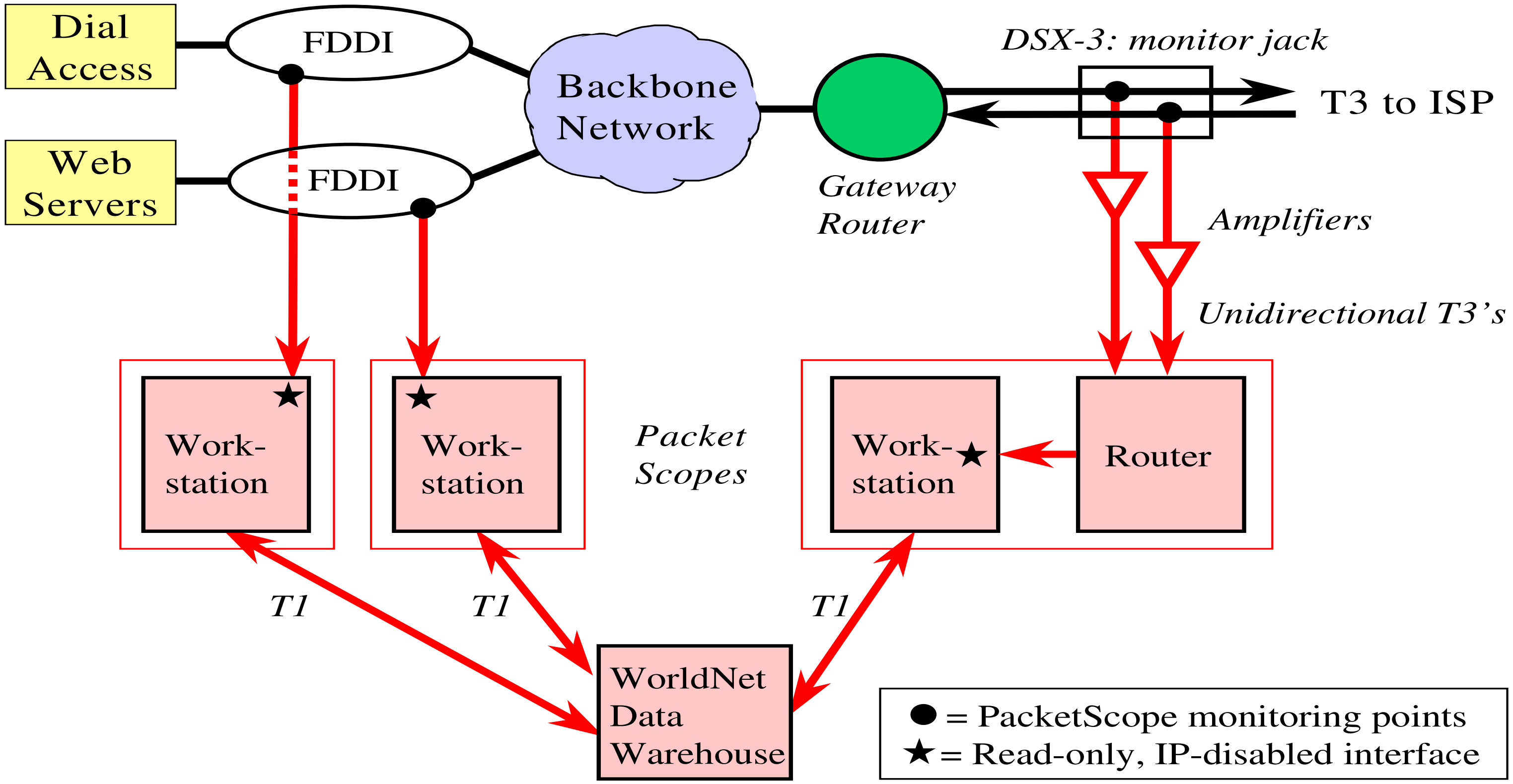
A few lines in extended common log format

```
18.17.1.15 - - [3/Feb/1999:00:00:01 +0000] "GET /today/weather.gif
    HTTP/1.0" 304 0 S_UA="MSProxy/1.0"
25.20.2.20 - - [3/Feb/1999:00:00:01 +0000] "GET /evil/eye/active.txt
    HTTP/1.1" 200 9120 S_UA="Mozilla/4.0 (compatible; MSIE 4.0; Windows NT)"
w-ok04.ath.com - - [3/Feb/1999:00:00:02 +0000] "GET /news/cpr_e.gif
    HTTP/1.0" 304 0 S_RU="http://www.shoodoo.op.org/dom/e1/2166"
    S_UA="Mozilla/2.0 (compatible; MSIE 2.1; AOL 3.0; Mac_PPC)"
25.20.2.20 - - [3/Feb/1999:00:00:02 +0000] "GET /evil/eye/active.txt
    HTTP/1.1" 200 9120 S_UA="Mozilla/4.0 (compatible; MSIE 4.0; Windows NT)"
24.15.8.15 - - [3/Feb/1999:00:00:03 +0000] "GET /events/goe.gif
    HTTP/1.0" 200 435 S_RU="http://www.shoodoo.op.org/events/vbe.html"
    S_UA="Mozilla/3.01 (Macintosh; I; PPC)"
```

Variations in what is logged and how

- Choice left to server configuration
- Format within fields vary (many formats of date string)
- Often ambiguous (especially time of request vs. response start vs. response end)
- Granularity (often at one second)

Trace gathering: details



Details

How is it processed?

Processing logs

Several steps in processing logs

- Parsing and cleaning
- Validating
- Altering representation
- Extracting subsets

Parsing and cleaning logs

- Logs are often dirty (not referring to URLs)
- Variable number of elements in a record
- Arbitrary length fields ($>$ 4K URL)
- Variety of formats for same element (e.g., 18 formats of date)
- Missing fields, dates in the past
- Data may be transient (IP addresses get reassigned)
- Syntactically and semantically dirty (embedded junk)

Need a safe way to parse and clean arbitrarily long, dirty logs quickly

Software's crucial role

- Scale of data to be processed
- Complexity
- Pre-processing/cleaning of data
- *ast* libraries and associated software

  regexp for URL parsing, cleaning entries
  sfio for reading abnormally large entries, no memory allocation
  libtime for handling 18 different time formats
  cql for generating statistics (write like awk/perl, run like C)

Software (source code!) available from
`http://www.research.att.com/sw/download`

Example of range and scale

Server Log (days)	Number of Requests	Number of Clients	Unique Resources
AIUSA (28)	180,324	7,627	1,102
Marimba (21)	222,393	24,103	94
Apache (49)	2,916,549	271,687	788
Sun (9)	13,037,895	218,518	29,436
EW3_4 (94)	6,273,792	79,623	2,638
EW3_6 (94)	6,281,303	102,454	784
EW3_18 (94)	3,902,302	58,839	279
EW3_60 (94)	45,903,646	481,628	2.031
IBM Nagano (1)	11,665,713	61,707	33,875
EventLog (1)	6,598,756	28,672	24,150
Superbowl (1)	9,869,394	212,549	421,078
DIMACS (90)	837,895	55,687	35,523

Towards a repository of clean and validated logs

- Most fields have ranges of values (response code, date etc.)
- Semantics checks can be automated
- Log entries can be cleaned, validated automatically
- Clean log repository notion created under W3C WCG auspices
- `http://purl.org/net/repository`

So how have I used Web data?

- Research –> data gathering –> analysis –> papers –> standards (via conference boondoggles)
- Create a set of reusable tools and processes
- Not be afraid of scale
- Want to have impact on the most widely used protocol on the Internet
- Work across network protocol layers
- Work on the more interesting applications (caching, traffic characterization)

Research projects using Web data

1. Web caching: improved cache validation, traffic reduction via piggybacking, proxy filters/server volumes for predicting access
2. HTTP Protocol improvement: Rate of change of Web pages
3. Addition of delta/compression to HTTP/1.1
4. HTTP/1.1 protocol compliance
5. Predicting HTTP/1.1 traffic from HTTP/1.0 traffic
6. Clustering Web clients

Web caching-piggybacking

- Introduced piggybacking: validating cached but possibly stale resources on needed connections (normal GET requests)
- Significantly reduce number of validation requests, improved freshness
- Pre-validation can aid proxy at no additional cost
- Packet traces at AT&T and converted traces to logs
- Demonstrated on many logs–but mainly 2 large ones (ATT, DEC)

USITS97, WWW-7

Web caching-volumes

- Use server's information on resources changing/accessed together
- Group resources into volumes and provide hints at volume level
- End to end information exchange: proxy's filters leading to tailored volume hints
- Be able to predict future accesses
- No protocol modification needed
- Used server logs from Sun, AIUSA, Apache, Marimba, and AT&T EW3

SIGCOMM'98, Infocom'99

Protocol improvement

- Rate of change of Web pages
- Examined rate of change of resources via packet traces and proxy log
- Content type, inter-access times, change frequency, types of changes
- Addition of delta/compression to HTTP/1.1
- Protocol compliance
- Predicting HTTP/1.1 traffic from HTTP/1.0 traffic

USITS97 , SIGCOMM'97, Global Internet'99, IETF'99

Clustering Web clients

- Where are requests coming from?
- How can client clustes be detected?
- Locate special clients (spiders, proxies) within the clusters?
- Use this information in real time?

Cluster detection and validation

- Use IP address in server (proxy) logs to cluster clients (servers)
- Use routing table snapshots—automatically obtained to create prefix/netmask entries table
- Match IP address using table to cluster
- 15 different logs, 99.99% of clients clustered with over 90% accuracy
- Includes real-time data
- Applicable to content distribution, proxy placement...
- Feasibility of technique, correctness, usefulness all depends on having diverse set of logs

Myths and realities

- Accurate data is possible
- There is privacy on the Web
- Popularity measures (Keynote, MediaMetrix etc.) are accurate
- Response popularity maybe more important than request, network wise
- Repository is a good idea, ongoing validated data from many sites will be hard
- Many products to analyze data syntactically but semantic analysis is spotty (start up!)