Methodology and tools to analyze DITL DNS data

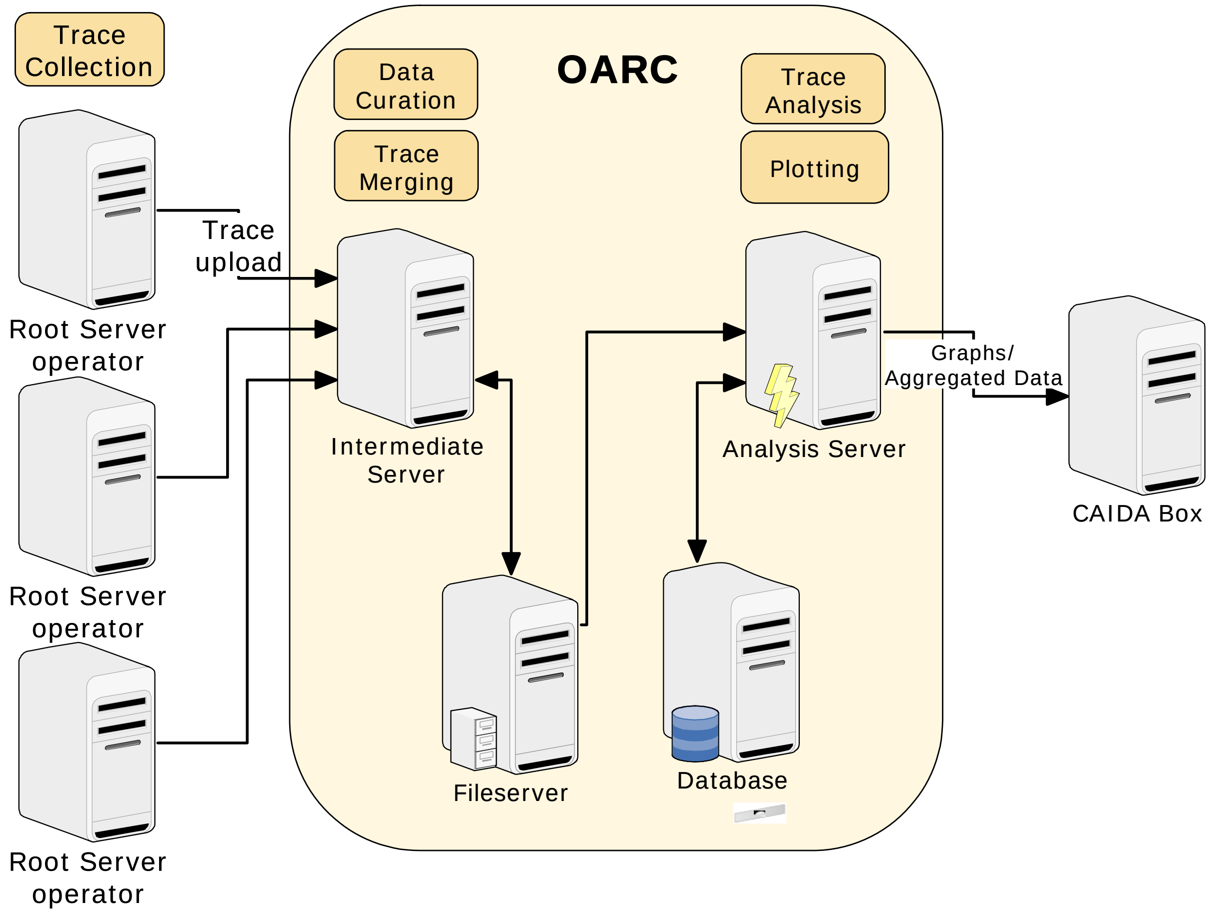
Sebastian Castro
secastro@caida.org

CAIDA

9th CAIDA/WIDE workshop – January 2008

Process overview

- Data collection
- Trace curation
- Trace merging
- Trace analysis
- Plotting



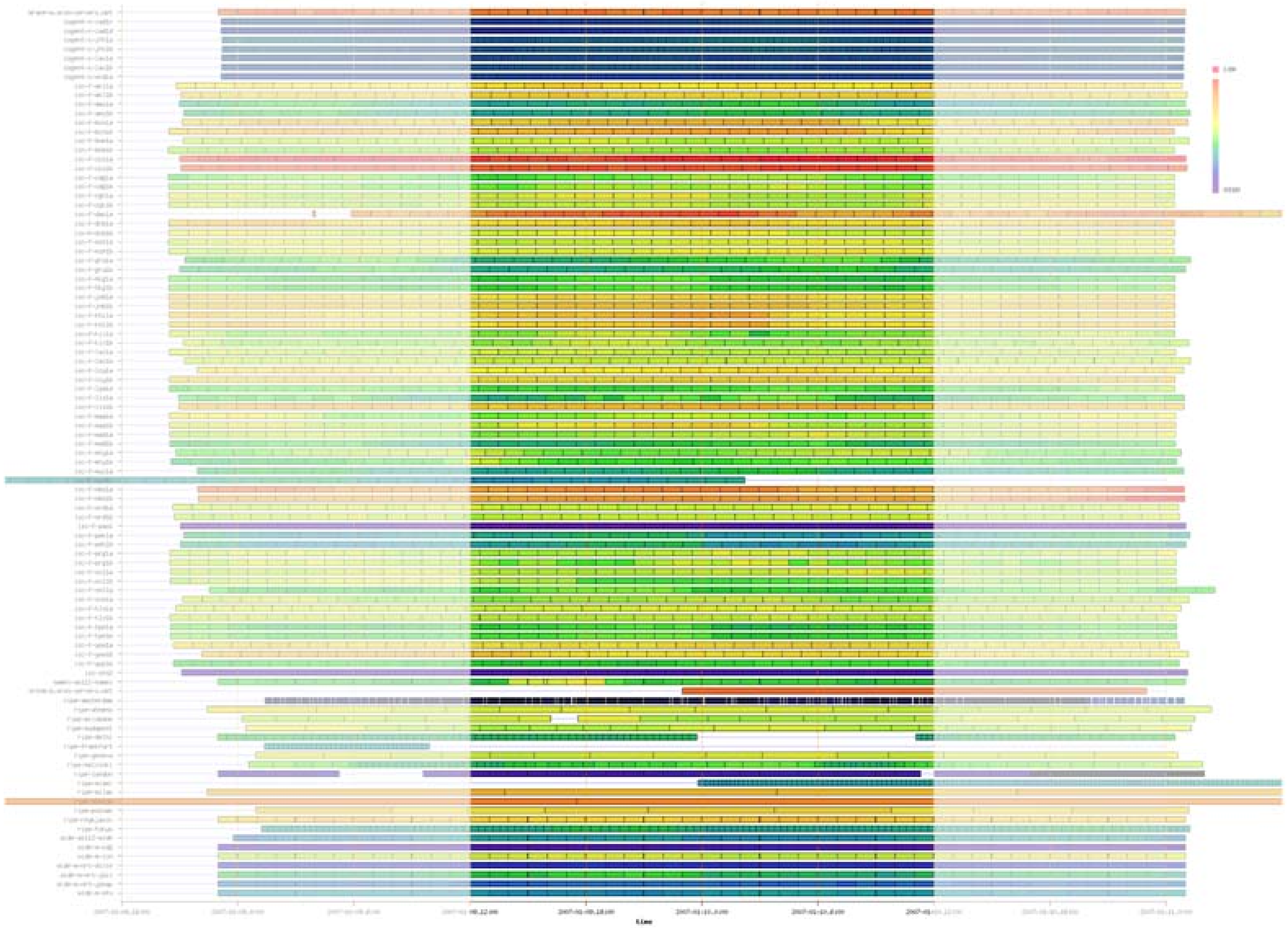
Data collection

- Done by each operator based on CAIDA recommendations
  - http://www.caida.org/projects/ditl/
- Using tcpdump, dnscap, etc and helper scripts
- All traces uploaded to OARC file server
  - All further processing done on OARC boxes due to data access restriction.

Data verification

- Verify trace completeness and integrity
  - Has missing pieces?
  - Truncated packets?
  - Truncated gzip files?
  - Check clock skew
  - Count DNS queries, responses, IPv4 packets, TCP, UDP, etc.
- Select the best dataset available
  - In terms of coverage
    - Defined as the number of packets seen versus the number of packets expected to seen.

Example of coverage



Trace merging

- Transform the original traces by
  - Homogeneous time intervals
    - 1-hour chunk
  - Correct clock skew (where known)
  - Translate destination addresses
    - All instances of the same root share the same IP, impossible to distinguish.
    - Some use private addresses internally.
    - Transform from 192.33.4.12(C-root) to 3.0.0.4 (3 represents C, 4 represents instance number)
  - Filter other traffic
    - DNS queries sent to other addresses on the same machine
    - Leave only queries
    - DNS traffic generated by the machine: zone sync traffic.
- To get one file per hour with all instances included

DNS analyses

- Analyses currently available
  - Client and query rates per instance
  - AS/prefix coverage per instance
  - Distribution of queries by query type
    - Global and deaggregated by root and instance
  - Node/cloud switching per client
  - Source port distribution
  - EDNS support (client and query), EDNS buffer size
  - Invalid queries
  - Recursive queries
  - RFC1918-sourced queries counter

Trace Analysis Tool

- Reads pcap and pcap.gz files
- Output as text file
    - SQL files to create tables and the data
    - Plain files with some stats
- C/C++ code
- Memory footprint
    - 300M – 6G
- Uses patricia trees to implement route table lookups

Database

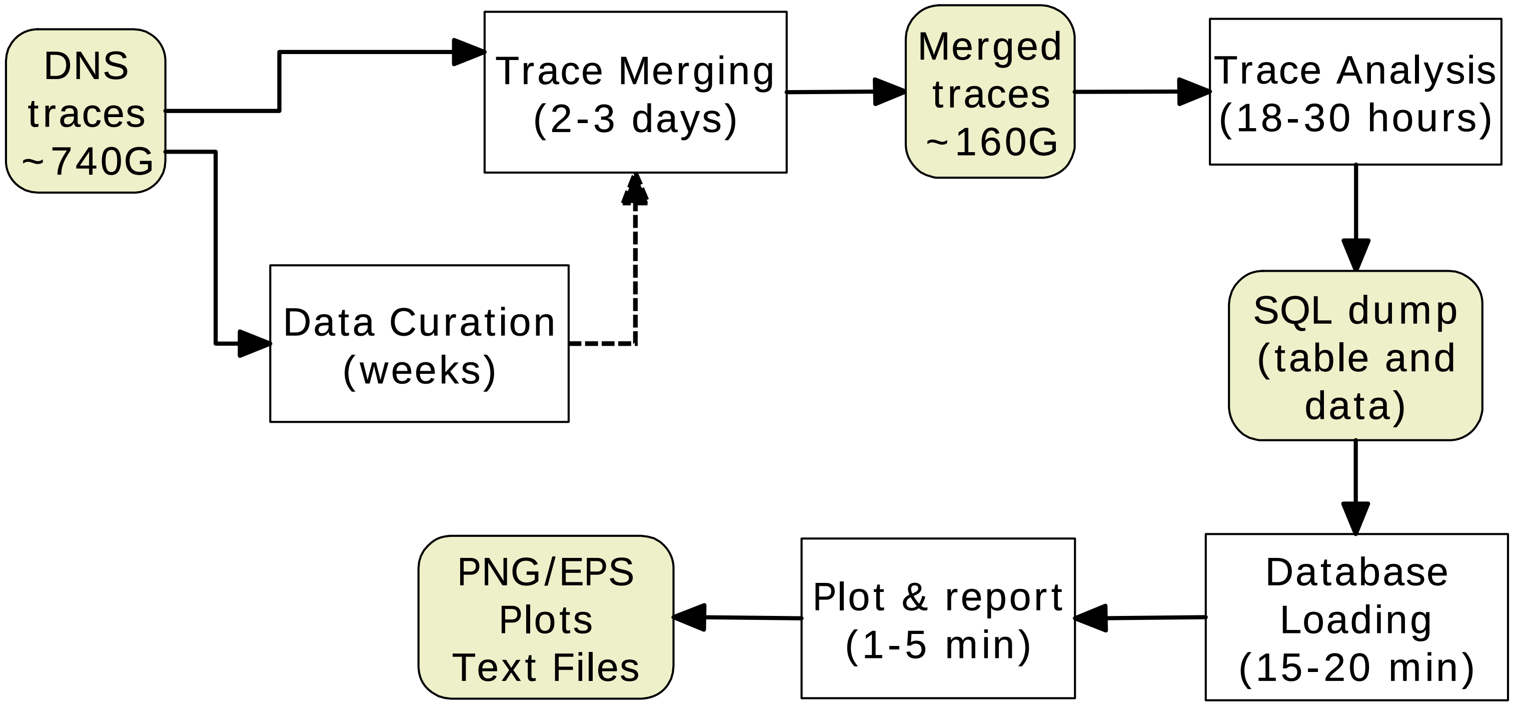
- PostgreSQL
  - Usually one table per analysis
  - Not much work on performance
  - Gave us some problems about table access control
- One database per dataset
  - Root traces 2007
  - Root traces 2006
  - ORSN 2007

Data presentation

- Some preprocessing/data aggregation done using Perl/AWK
- Graph generated using ploticus
- Group things could be easily done

Process example

DITL 2007 analysis flow example



Recent improvements

- Have better performance
  - Replaced *map* with *hash_map* (unordered associative arrays) for a 40% performance gain
- Simpler selection of analysis to run
  - Using command line
- A object-oriented design
  - More organized code
  - Allowing others to add analyses

What’s next

- Add new analyses
  - Daily patterns by query type
  - Locality of queries by TLD
  - Improve some criteria on the invalid query classification
  - IPv6 related traffic (queries and packets)
  - … put your desired analysis here …

Conclusions

- Having tools and procedures to collect and analyze the data makes things easier.
  - Allowed us to make comparisons between 2006 and 2007 pretty straightforward
- Current tools covers the basics
  - Clearly subject to be improved and extended
  - Performance could become an issue with larger datasets.