DNSql : Processing Massive DNS Collections (abstract)

Stephen Herwig, Dave Levin, Bobby Bhattacharjee, Neil Spring
University of Maryland, College Park

The University of Maryland operates DNS's D-root nameserver. For robustness, D-root uses anycast to distribute its service across 98 replicas throughout the world [1]. For each hour, each replica collects a sample (the first 90 seconds or more of each 10 minute period) of DNS traffic, and stores the sample as a compressed pcap. Collection started in October 2014; the total size of collected traffic is 78 TB, with data added at a rate of roughly 140 GB per day.

Our goal is to develop scalable and flexible techniques to analyze months' or years' worth of collected traffic. While datasets such as RSSAC-002 [4] provide daily aggregate statistics such as traffic volume, aggregate values are not always useful inputs to algorithms for discovering traffic behaviors and anomalies. For example, analysis of catchment stability or suspicious activity often require features constructed at the packet level, such as measurements of query and source diversity, or client switches between replicas. Our initial methods to analyze long time spans used tools similar to *dnstop* [5] to process pcaps serially and compute features for downstream applications. Processing a month's worth of D-root data in this fashion took days to complete, and pre-defining the features limited subsequent modification of the analysis design. Attempts to import the pcap data to a single PostgreSQL database, so as to support ad hoc analysis, were too slow to keep pace with traffic collection.

Several inspiring works processed massive DNS collection by transforming the collected traffic to an intermediate representation (IR). Plonka et. al. [2] developed data structures for storing IP addresses and query names to facilitate clustering DNS traffic. van Rijswijk-Deij et. al. [3] converted active DNS measurements to Apache Parquet, a columnar storage format suited for Hadoop. We seek an IR that makes fewer assumptions on the types of analyses and the analytic framework.

We present our initial design of a scheme for converting the pcaps to database shards, and performing MapReduce-style queries over the shards. For this scheme to be beneficial, the MapReduce queries must be faster than equivalent queries on the hypothetical aggregate database, and conversion of pcaps to shards should be faster than aggregating the pcaps as a single database. Also, ideally the shards should be smaller than the pcaps.

For our implementation, we developed a tool, *dnsqlite3c*, to convert a compressed pcap to an SQLite3 database, with the conversion being only slightly slower than a baseline approach of reading the pcap with a *zcat* and *tcpdump* pipeline. As DNS responses echo the original request, and as TCP represents roughly 1% of queries, *dnsqlite3c* currently only parses UDP responses. The created database contains two tables. The first table has a row for each DNS response in the pcap, with fields such as the client's IP address, the query name (qname), and the result code (rcode). *dnsqlite3c* creates standard SQLite3 B-tree indices over the client's IP address field and the qname field, by default. The second table is a precomputation of the number of queries-per-second (QPS), a useful metric for traffic volume.

Due to the exclusion of DNS queries, as well as many link layer, IP, and UDP header fields, a shard is roughly 8× smaller than the uncompressed version of the pcap from which it was created. Shards compress well (roughly 2.3 compression factor), but since our main priority is speed rather than storage, we choose to leave the shards uncompressed.

For our initial analysis of shard creation and query times, we focused on the traffic for one replica – College Park, Maryland (CPMD) – for March 2015. CPMD is a globally visible replica. CPMD does not sample, but rather collects traffic for the entire hour. The total size of the compressed pcaps for CPMD for March 2015 is 472 GB, representing over 2.6 billion UDP DNS responses.

In order to create many shards quickly, we create shards in parallel via a process pool; each process in the pool converts a separate pcap. Compared to creating a single aggregate SQLite3 database for CPMD March 2015, creating shards in parallel is roughly an order of magnitude quicker (9.4h vs. 1h).

We developed the MapReduce application using Python's *multiprocessing* library. We also developed two extension libraries in C: one for SQLite3 and one for Python. The SQLite3 extension provides functions for manipulating IP addresses and qnames, as well as hashing strings; the Python library implements a bit array. Combined, the extensions allow efficient representation of sets of IP addresses and hashed qnames.

We compared the speed of the MapReduce queries to analogous queries on the aggregate CPMD March 2015 database using queries of roughly two types: querying the frequency of a table field and querying the distinct values for a field. Using this approach, the queries involving client IP addresses were 5–7× faster than queries on the aggregate database, and queries involving hashed (instead of full) qnames were roughly 2× faster.

Given the opportunity to query traffic at the packet level (rather than the aggregate), what additional questions can we now ask? What opportunities are there to 'join' the shard tables with secondary sources, such as IP geolocation and BGP routing datasets, and what are efficient methods for exposing such datasets to the queries? Does this overall scheme scale to multiple root and TLD nameservers, and what privacy challenges do the non-aggregated databases pose?

References

[1] Matthew Lentz et al. “D-mystifying the D-root Address Change”. In: *Proceedings of the 2013 Conference on Internet Measurement Conference*.

[2] David Plonka and Paul Barford. “Context-aware Clustering of DNS Query Traffic”. In: *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*.

[3] Roland van Rijswijk-Deij et al. “The Internet of Names: A DNS Big Dataset”. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*.

[4] *RSSAC. Advisory on Measurements of the Root Server System*. Nov. 2014.

[5] D. Wessels. *dnstop*. URL: `http://dns.measurement-factory.com/tools/dnstop`.