Atlas & Ark data in Google BigQuery

sds@caida.org
sds@ripe.net

September 5th, 2019
motivation

• Atlas and Ark both collect a lot of data
• “big data” is now old-hat and commodity platforms are pretty well established
  • m-lab uses bigquery
  • censys uses bigquery
• they have bigger datasets than we do
motivation

• the RIPE NCC is exploring putting data into BigQuery
• using the platform and looking for concrete uses against RIPE NCC data can help prove its worth
  • or in contrast, can confirm its expense
• element of testing the platform and the data pipeline, not just doing analysis
  • feedback loop: “is this useful?”
• but this was all new to me
  • for my AIMS2019 talk I had ~3 days experience with bigquery
BigQuery

• ‘serverless’, ‘real-time’, ‘analytics’ ‘data-warehouse’ ‘platform as a service’
  • no ops
  • queries are fast, but not immediate
  • SQL-interface on top of datasets
  • the ability to just throw in tera/petabytes of data
goals

- Multiple:
  - understand how to use this platform and what work it is good for
  - build use-cases:
    - operationally-motivated work
    - research-motivated work
  - throw raw data in, and operate from first-principles
“ Doesn't this cost money?”

- yes!
- but for many studies, it’s likely to be cheaper than the capital + operational spend on rolling your own
- and currently the NCC provides data, not compute
Atlas traceroute data was being funnelled in here before I started working on it in April.

Can we get to the point of answering real questions?
Atlas measurements

Atlas, total number of results collected per hour

# results

Date Range: 2019-08-01 to 2019-09-01

- IPv4
- IPv6

$0.21
Atlas runs multiple classes of measurements:

1. **built-in measurements**; [https://atlas.ripe.net/docs/built-in/](https://atlas.ripe.net/docs/built-in/)
   - all probes ping/traceroute/dns to DNS root servers & Atlas anchors

2. **topology sweep**: UDP + ICMP traceroutes

3. **anchoring measurements**
   - full-mesh ping/traceroute between all anchors
   - random collection of other probes ping/traceroute each anchor

4. **user-defined measurements**: public + private
   - msm_id >= 1,000,000
   - is_public = [true|false]
Atlas measurements

- What fraction of the archived RIPE Atlas data was generated from each of the measurement categories above?
- How has this evolve?
multiple sources

- atlas_traceroute
- anchoring_msms
- meta_20190827

- tagged_atlas_traceroutes

- adjacencies_atlas_all
Atlas measurements

$1.39; $3.41 for a prior query for an intermediate table
targets per hour

Atlas, unique targets per hour

$2.17^*$
user measurements

$1.39$ (data from same queries as earlier slide)
Atlas (user-defined) targets

Atlas, unique user-defined targets per hour

$2.17^*$
Ark data

- IPv4

- IPv6
Ark data

Ark, total number of results collected per hour

$0.03
Atlas measurements

- Size of dataset
  - in bigquery,
    41.5GB/day (IPv4), and
    18.5GB/day (IPv6)
How long does it take Atlas to complete IPv4 and IPv6 topology sweeps?
Atlas (topology) targets
Atlas measurements

How rapidly does the set of user-defined targets grow?
Atlas (user-defined) targets

Atlas, user-defined target accumulation [2019-07-01 -- 2019-08-31]

$0.56
Atlas measurements

• Aim: characterise the diversity in the IPv4 and IPv6 topology data collected from each platform, in terms of vantage points, targets, and intermediate hops, along the following granularities:
  • BGP-routed prefixes,
  • networks (AS-level),
  • network types, and
  • geographic coverage?

• What do the platforms observe differently from each other?
responding addresses

- start to look at the diversity in the collected IPv4 and IPv6 topology data
- both platforms
Atlas responding addrs


$14.40
Atlas responding addrs

1 day  Atlas, y IPs observed by x probes in time unit  1 week

$1.93
Ark responding addrs
Prefix-matching

- BigQuery doesn’t understand what a subnet is
- It does understand *ranges*, and IP addresses are just numbers
Longest-prefix Matching

• Matching is relatively easy
• Taking a set of matches and choosing the one with the longest prefix: trickier (in an SQL language)
  • but definitely achievable
WITH ips AS

(SELECT distinct rh.from as ip
FROM "data-test-194508.prod.traceroute_atlas_prod", unnest(hops) as h, unnest(resultHops) as rh
WHERE af = 6)
,

-- Select prefixes from BGP table here
filtered_bgp_table AS

(SELECT distinct prefix, start, foo.end, masklen, origin
from "data-test-194508.adg_test.rtt_20190421_1600" foo
WHERE af = 6)
,

-- Meat of the join happens here.
-- JOIN will lead to combinatorial explosion so minimise first by matching
-- on a /29, then match on the value actually in the BGP table.
-- Will result in multiple rows if #matches > 1
-- Will result in NULLs if #matches == 0
matched_results AS (
SELECT ips.*, bgp.*
FROM ips
LEFT JOIN filtered_bgp_table bgp
ON NET.IP_TRUNC(NET.IP_FROM_STRING(ip), 29) = NET.IP_TRUNC(bgp.start, 29)
AND (NET.IP_FROM_STRING(ip) BETWEEN bgp.start AND bgp.end)
)
,

-- For each IP:origin, retain the longest prefix only
matched_results_ips AS (  
WITH a AS (  
SELECT distinct ip, max(masklen) as m  
FROM matched_results  
group by ip  
)
SELECT distinct b.*
FROM matched_results b
JOIN a
ON (b.ip = a.ip and b.masklen = a.m) or (b.prefix is null)
),

-- Aggregate multiple origins into one array
aggred_asns AS (  
-- IGNORE NULLS here is important in the case that there is no match in this BGP table  
SELECT ip, ARRAY_AGG(origin IGNORE NULLS) as origins
FROM matched_results_ips
GROUP BY ip
)
,

-- Spit out everything
SELECT *
FROM aggred_asns
Platform Comparison?

- Can we just start by looking at:
  - IPs observed
    - day/week
  - IP-level adjacencies observed
    - day/week
  - AS-level adjacencies observed
    - day/week
Observed per day:

<table>
<thead>
<tr>
<th></th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Atlas</td>
<td>Ark</td>
</tr>
<tr>
<td>Addr</td>
<td>793,342</td>
<td>4,811,733</td>
</tr>
<tr>
<td>/24s</td>
<td>394,368</td>
<td>2,344,294</td>
</tr>
<tr>
<td>ASNs</td>
<td>45,154</td>
<td>36,589</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Atlas</td>
<td>Ark</td>
</tr>
<tr>
<td>Addr</td>
<td>164,989</td>
<td>282,238</td>
</tr>
<tr>
<td>/48s</td>
<td>30,787</td>
<td>56,501</td>
</tr>
<tr>
<td>ASNs</td>
<td>8,305</td>
<td>10,165</td>
</tr>
</tbody>
</table>

$2.15$
Observed per week:

<table>
<thead>
<tr>
<th></th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Atlas</td>
<td>Ark</td>
</tr>
<tr>
<td>Addr</td>
<td>1,117,248</td>
<td>41,824,271</td>
</tr>
<tr>
<td>/24s</td>
<td>496,282</td>
<td>3,632,743</td>
</tr>
<tr>
<td>ASNs</td>
<td>45,160</td>
<td>36,589</td>
</tr>
<tr>
<td></td>
<td>239,417</td>
<td>1,792,204</td>
</tr>
<tr>
<td>/24s</td>
<td>34,836</td>
<td>143,445</td>
</tr>
<tr>
<td>ASNs</td>
<td>8,307</td>
<td>10,165</td>
</tr>
</tbody>
</table>
Adjacencies

- This is trickier, but interesting
- The language operates on rows and columns, but rows are distinct
- Traceroute results as rows: not useful
- But, arbitrary code is acceptable
## Adjacencies

<table>
<thead>
<tr>
<th>ipFrom</th>
<th>dstAddress</th>
<th>dstName</th>
<th>startTime</th>
<th>af</th>
<th>parsid</th>
<th>msmid</th>
<th>prbid</th>
<th>hops.hop</th>
<th>hops.resultHops.err</th>
<th>hops.resultHops.rtt</th>
<th>hops.resultHops.from</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.805999942779541</td>
<td>2001:67c:6ec:201:145:220:0:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.59600001575362562</td>
<td>2001:67c:6ec:201:145:220:0:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.768999938011169</td>
<td>2001:7f8:1::a500:6939:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9020000100135803</td>
<td>2001:7f8:1::a500:6939:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.7400000095367432</td>
<td>2001:7f8:1::a500:6939:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9.35099983215332</td>
<td>2001:470:0:431:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9.307000160217285</td>
<td>2001:470:0:431:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13.701000213623047</td>
<td>2001:470:0:410:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13.61999988559082</td>
<td>2001:470:0:410:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>73.25399780273438</td>
<td>2001:470:0:440:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>73.4219970703125</td>
<td>2001:470:0:440:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>73.26699829101562</td>
<td>2001:470:0:440:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>73.39399719238281</td>
<td>2001:470:0:20a:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>73.28099822998047</td>
<td>2001:470:0:20a:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>73.399000207519531</td>
<td>2001:470:0:20a:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>95.3209914550781</td>
<td>2001:470:0:270:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>92.66999816894531</td>
<td>2001:470:0:270:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>95.2229995727539</td>
<td>2001:470:0:270:2</td>
<td></td>
</tr>
</tbody>
</table>
CREATE TEMP FUNCTION GetHops(srcAddr STRING, json_str STRING)
RETURNS ARRAY<STRUCT<
    a STRING,
    b STRING>>
LANGUAGE js
AS ""
function keysrt(key) {
    return function(a,b){
        if (a[key] > b[key]) return 1;
        if (a[key] < b[key]) return -1;
        return 0;
    }
}
var row = JSON.parse(json_str);
var out_array = [];
row.sort(keysrt('probeTtl'));
for (var i = 0; i < row.length; i++) {
    // [snip]
    // logic
    // [snip]

    return out_array;
    "";

WITH filtered_traceroutes AS (
    SELECT startTime as start_time, srcAddress as src_addr, hops
    FROM "data-test-194508.caida.ark_traces_ipv6"
    WHERE startTime > "2019-08-01" AND startTime < "2019-09-01"
    AND srcAddress != ""
)

SELECT start_time, src_addr, GetHops(src_addr, to_json_string(hops)) as h
FROM filtered_traceroutes
## Adjacencies*

<table>
<thead>
<tr>
<th></th>
<th>Time Stamp</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
<th>Node 6</th>
<th>Node 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>2019-07-12 01:00:00 UTC</td>
<td>192.168.1.10</td>
<td>188.176.98.1</td>
<td>83.88.1.81</td>
<td>198.32.118.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>198.32.118.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.237.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.229.225</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.235.234</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.227.138</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.223.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>2019-07-12 01:00:00 UTC</td>
<td>192.168.1.10</td>
<td>188.176.98.1</td>
<td>83.88.1.81</td>
<td>198.32.118.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>198.32.118.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.231.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>2019-07-12 01:00:00 UTC</td>
<td>192.168.1.10</td>
<td>188.176.98.1</td>
<td>83.88.1.81</td>
<td>198.32.118.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>198.32.118.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.231.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.229.225</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.235.234</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.227.138</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209.148.223.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64.71.240.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>69.63.243.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adjacencies*

- IPv4+IPv6 level forward-adjacencies (1 day):
  - Atlas: 2,603,418
  - Ark: 6,196,998
  - Intersection: 921,464

- AS level forward-adjacencies (1 day):
  - Atlas: 102,766
  - Ark: 76,741
  - Intersection: 51,401

* no external validation on any of these yet!

$0.50? adjacency computation; $0.25? longest-prefix for 1 day; $0.25 (AS mapping computation)
back to money

1–31 Aug 2019
BigQuery Analysis: 86.647 Tebibytes
(Source: bq-test [bq-test-237918])

US$428.24

- 95,269,384,011,907 bytes of data queried
- $1 $= 205GB (at that rate)
- free tier offers:
  - one-time $300 free credit (~60TB of queries)
  - each month: 1TB free queries
back to money

- can you accidentally spend a lot of cash?
  - *yes of course*
- you can also set up spending alerts
- [https://edu.google.com/programs/credits/research/](https://edu.google.com/programs/credits/research/)
- [https://cloud.google.com/free/](https://cloud.google.com/free/)
```sql
SELECT DATETIME_TRUNC(DATETIME(startTime), HOUR) as ts, COUNT(DISTINCT dstAddress) as count
FROM data-test-194508.caida.ark_traces_ipv6
GROUP BY ts
ORDER BY ts
```

This query will process 13.8 GB when run.
### Table Info

| Description       | None
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Table ID</td>
<td>data-test-194508:caida.ark_traces_ipv6</td>
</tr>
<tr>
<td>Table size</td>
<td>663.23 GB</td>
</tr>
<tr>
<td>Number of rows</td>
<td>406,533,406</td>
</tr>
<tr>
<td>Created</td>
<td>30 Jul 2019, 01:46:35</td>
</tr>
<tr>
<td>Table expiry</td>
<td>Never</td>
</tr>
<tr>
<td>Last modified</td>
<td>4 Sep 2019, 01:32:22</td>
</tr>
<tr>
<td>Data location</td>
<td>EU</td>
</tr>
</tbody>
</table>
direction

this actually looks pretty feasible as an ongoing project
direction

- want: eventually to offer public access to data
  - maybe the way measurement-lab does it
  - maybe the way censys.io does it
  - maybe via Google’s public dataset program
    - tbd!
- current project will be wrapped
  - new project will be set up, with clearer user permissions, better-aligned schemas
future

• I’m back at the NCC from October 1st
• This project will iterate
  • More measurement types: ping, etc
  • Sampled historical/backfilled data
  • More consistent Ark data!
  • ITDK
  • IXPs dataset
  • geo dataset(s)
contact

sds@ripe.net

examples

https://gist.github.com/sdstrowes/c2e36b446cee4b8a5ff1e62d1c14e2e0