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

$$BigQuery$$

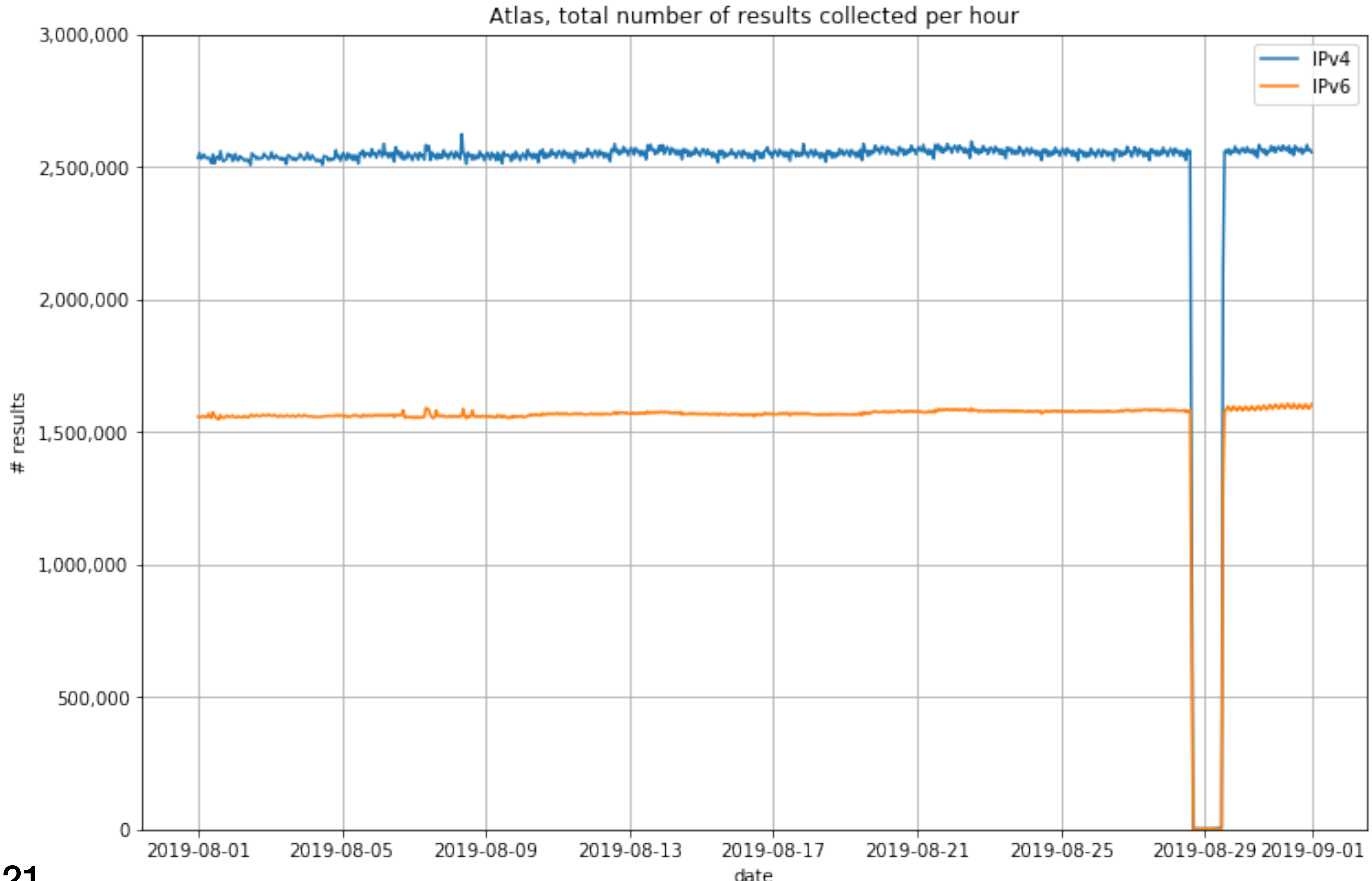
- “Doesn't this cost money?”
  - yes!
  - but for many studies, it’s likely to be cheaper than the capital + operational spend on rolling your own
  - and currently the NCC provides *data*, not *compute*

Atlas measurements

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



$0.21

Atlas measurements

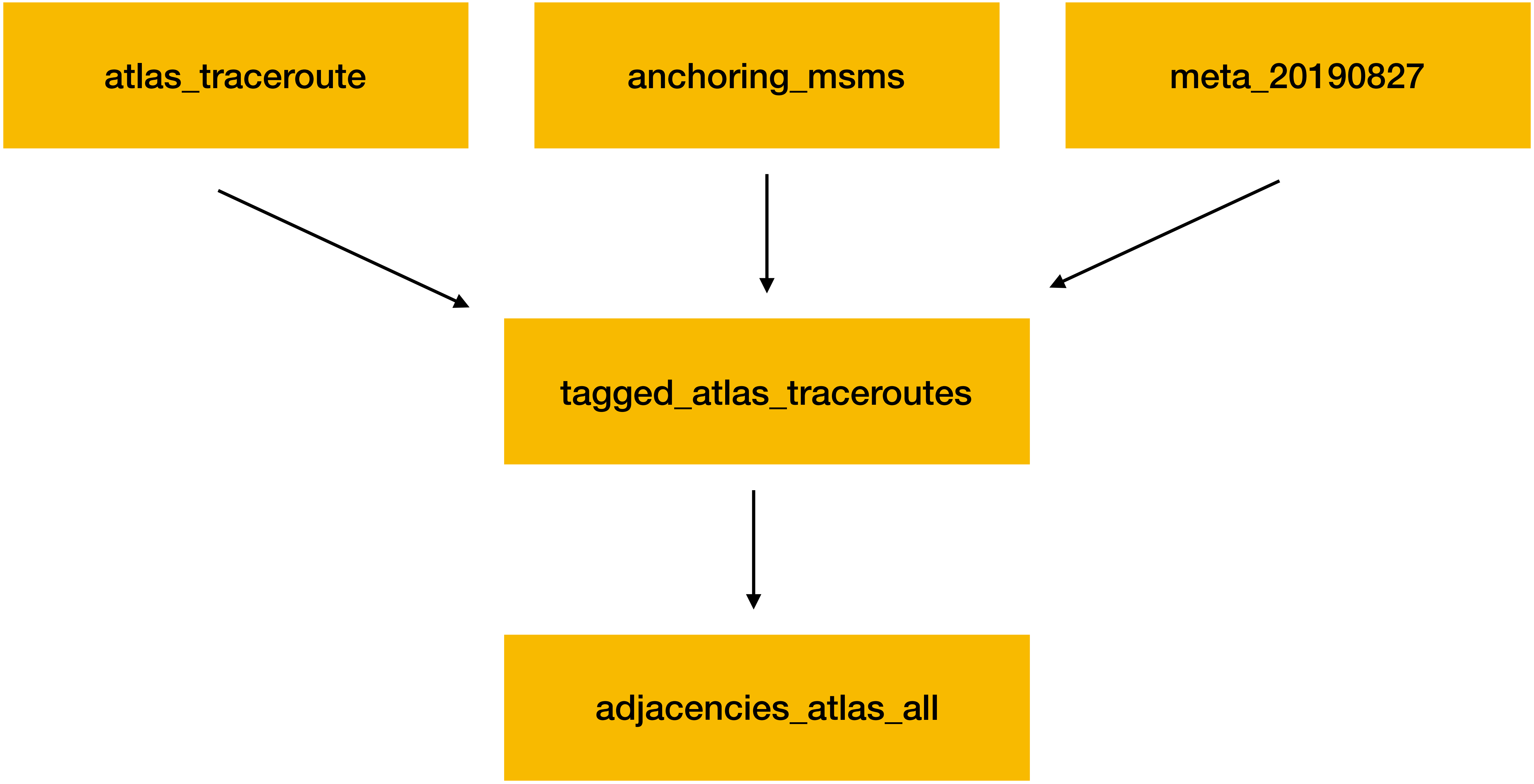
Atlas runs multiple *classes* of measurements:

1. **built-in measurements; 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]

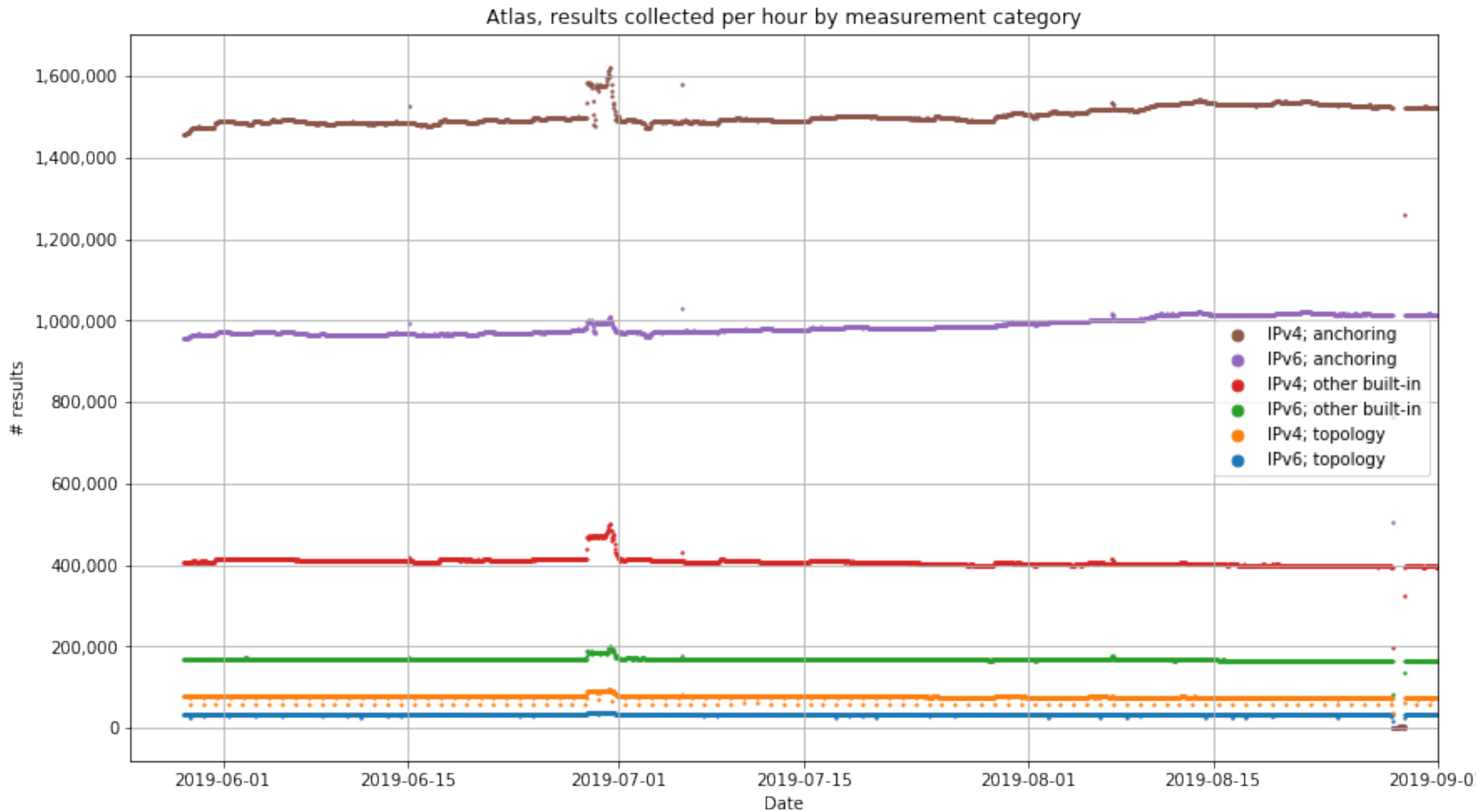
Atlas measurements

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

multiple sources

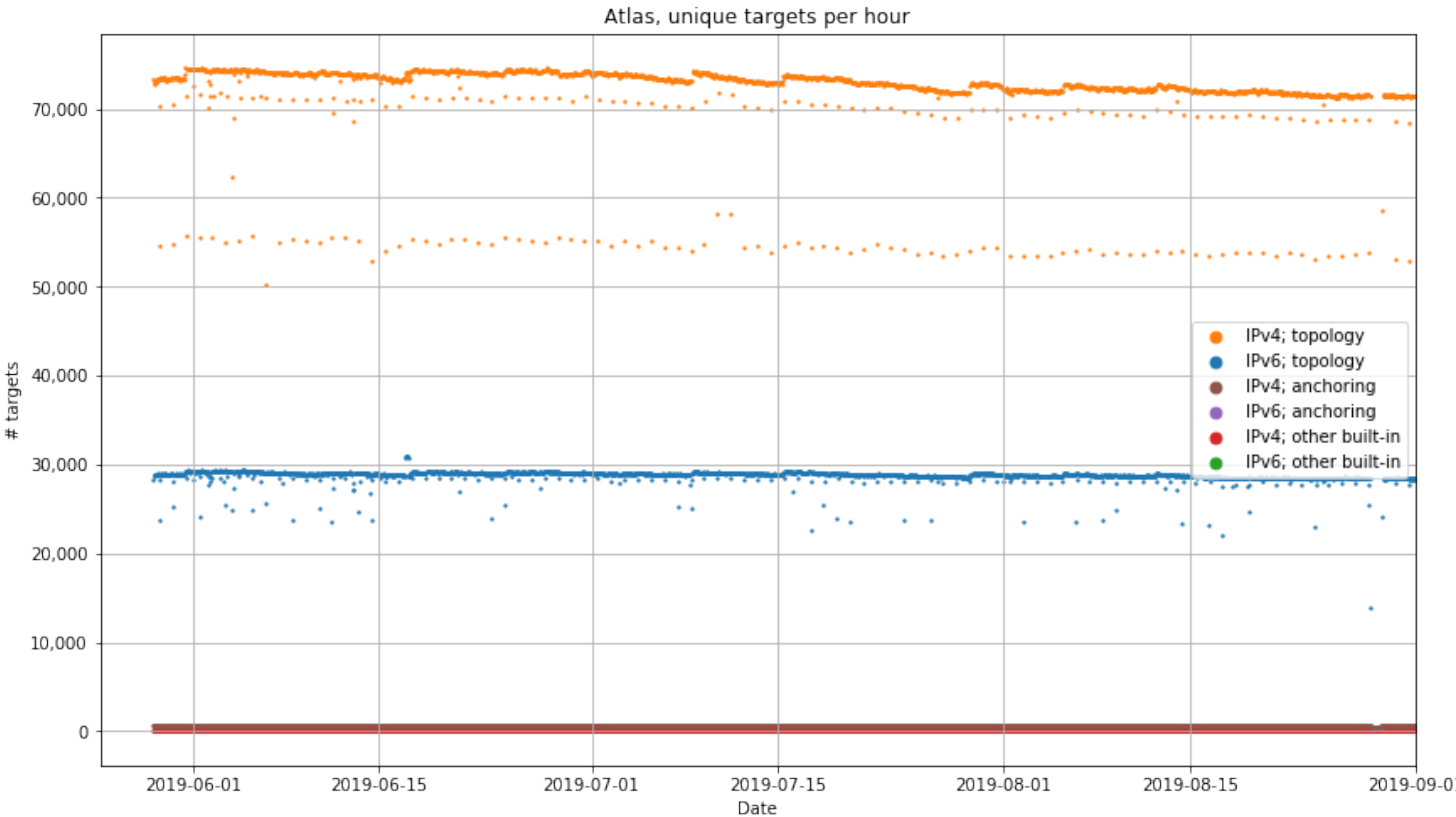


Atlas measurements



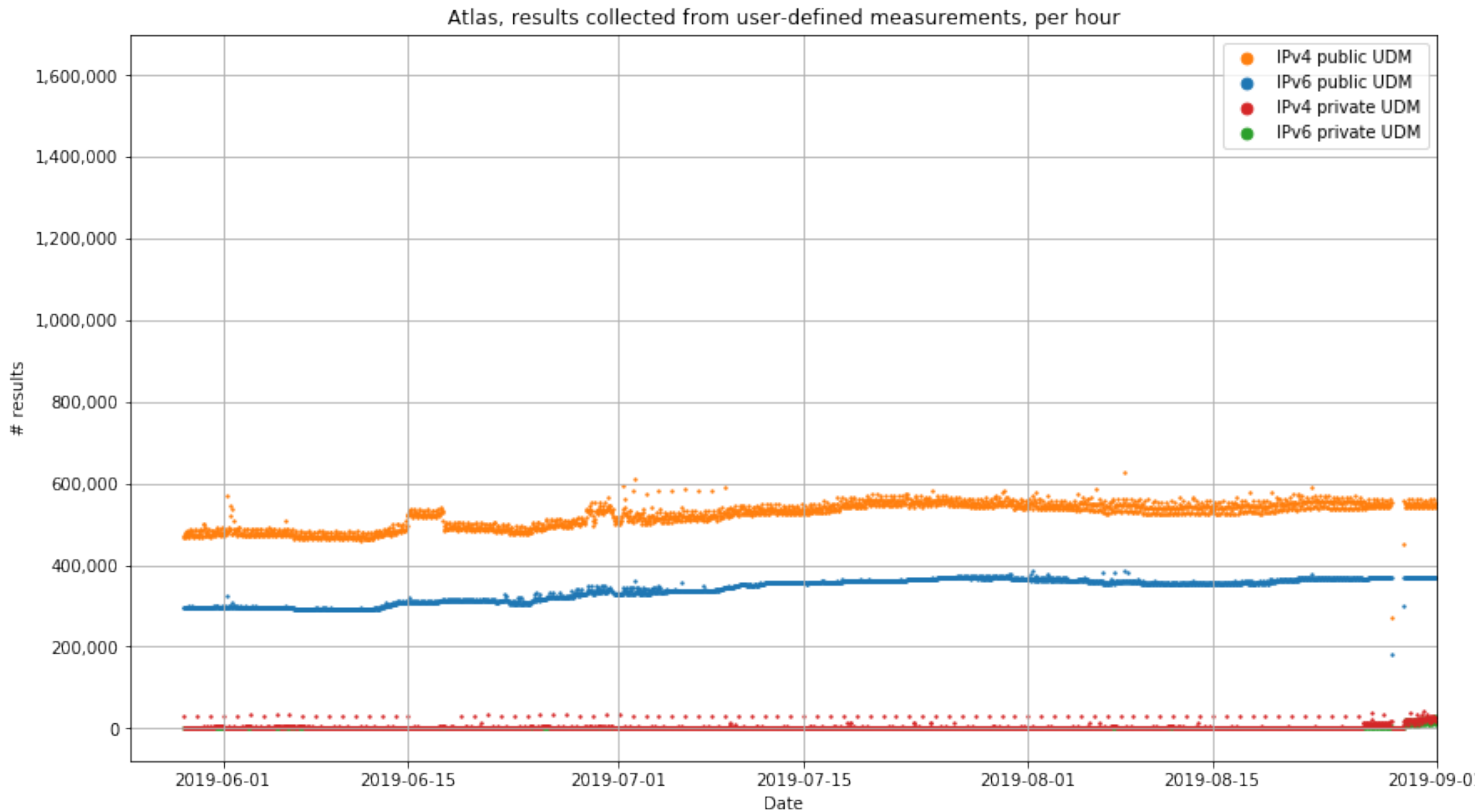
$1.39; $3.41 for a prior query for an intermediate table

targets per hour



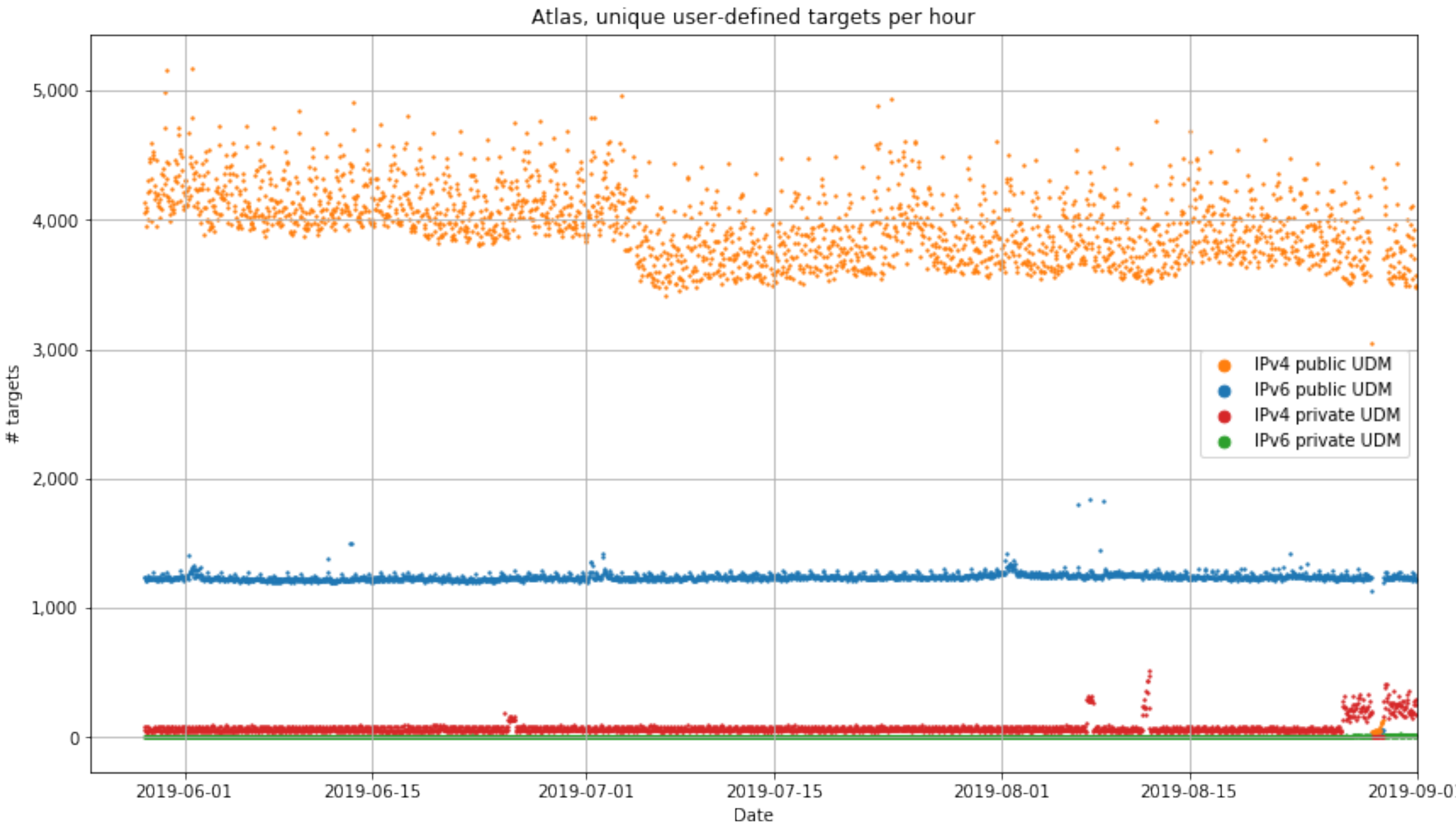
$2.17*

user measurements



$1.39 (data from same queries as earlier slide)

Atlas (user-defined) targets

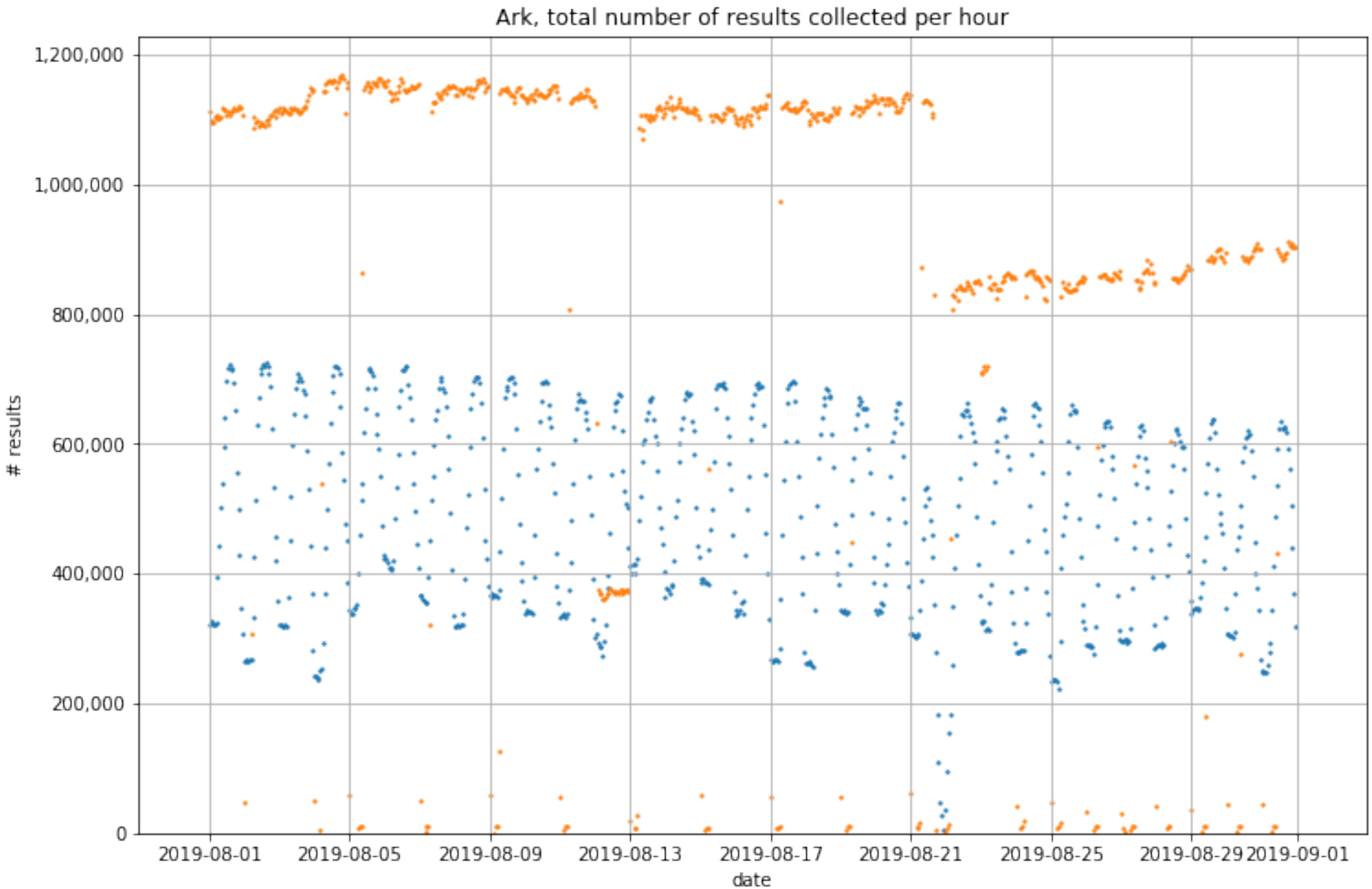


$2.17*

Ark data

- IPv4
  https://data.caida.org/datasets/topology/ark/ipv4/probe-data/team-1/daily/
- IPv6
  http://data.caida.org/datasets/topology/ark/ipv6/probe-data/

Ark data


Ark, total number of results collected per hour
results
1,200,000
1,000,000
800,000
600,000
400,000
200,000
0
2019-08-01
2019-08-05
2019-08-09
2019-08-13
2019-08-17
2019-08-21
2019-08-25
2019-08-29
2019-09-01
date

$0.03

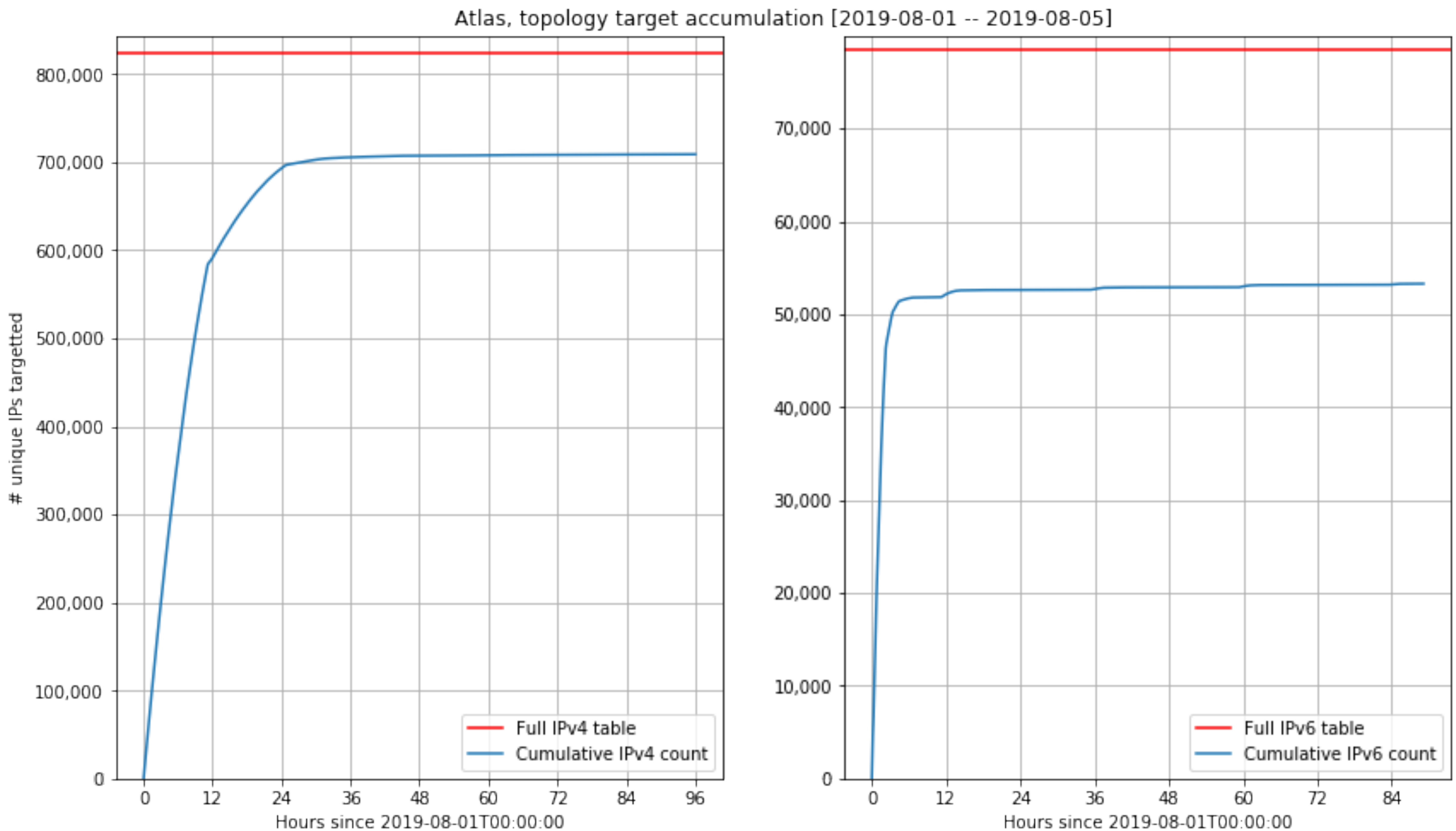
Atlas measurements

- Size of dataset
  - in bigquery,
    41.5GB/day (IPv4), and
    18.5GB/day (IPv6)

Atlas measurements

How long does it take Atlas to complete IPv4 and IPv6 topology sweeps?

Atlas (topology) targets

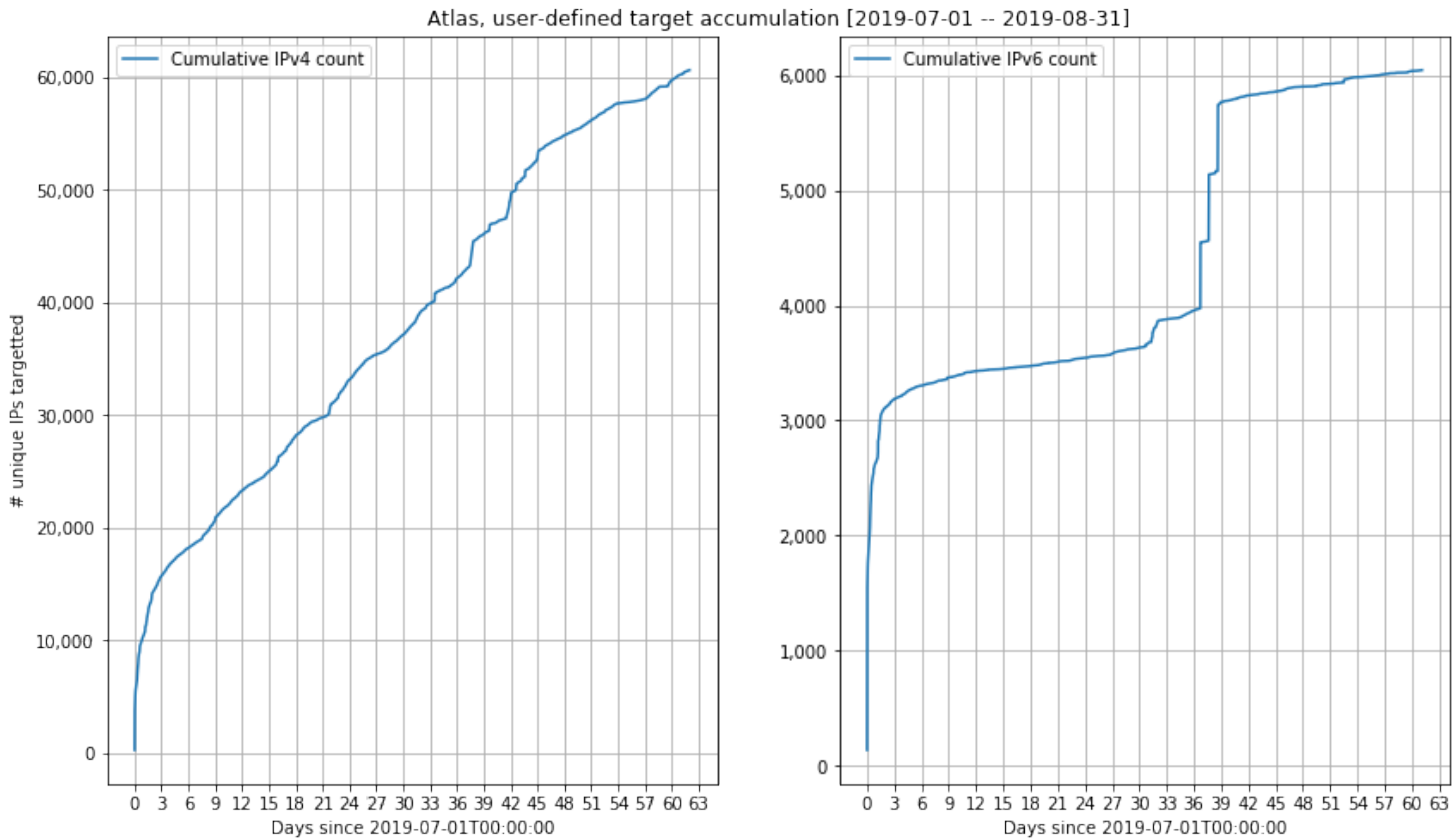


$0.12

Atlas measurements

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

Atlas (user-defined) targets



$0.56

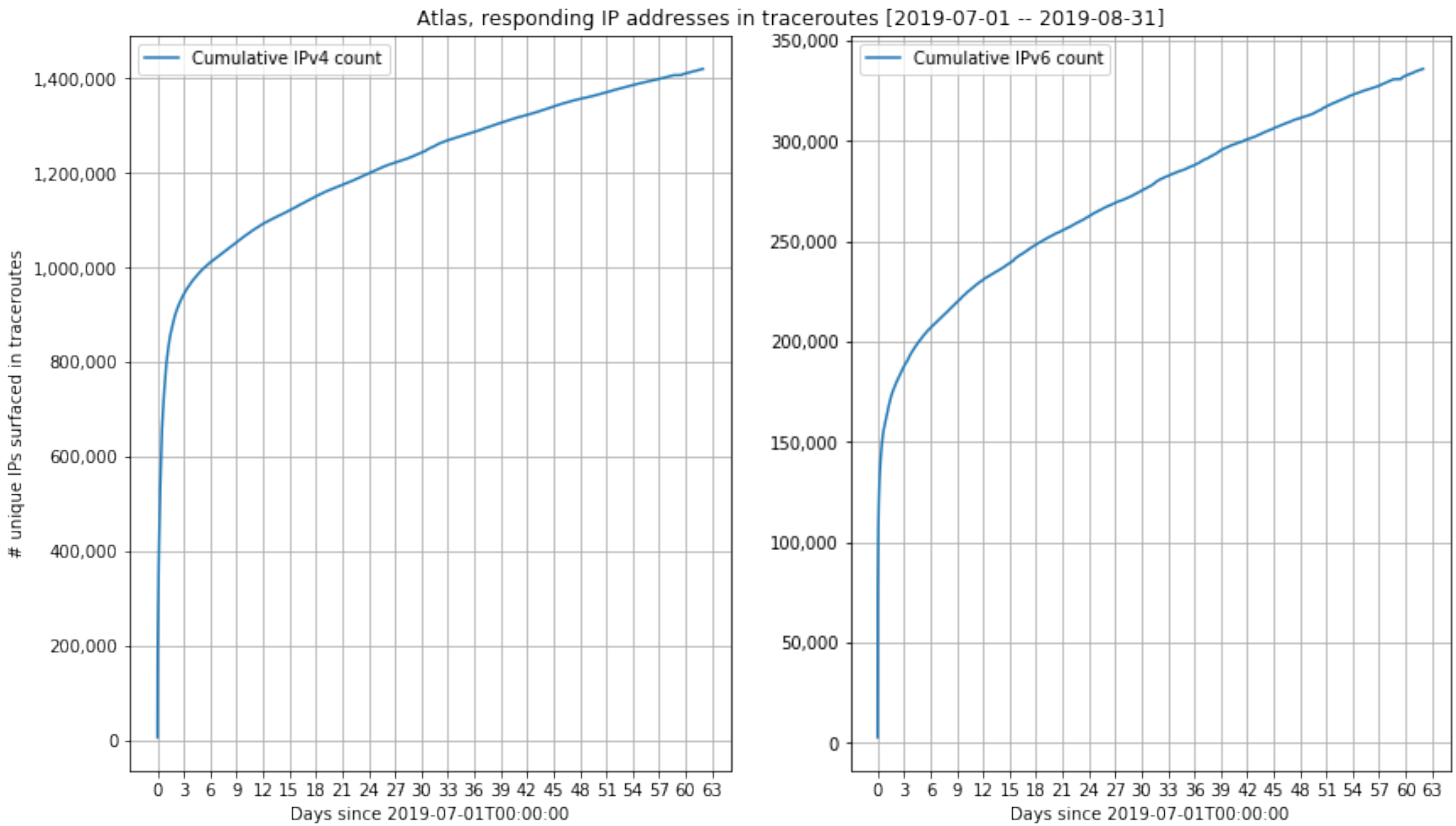
Atlas measurements

- 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

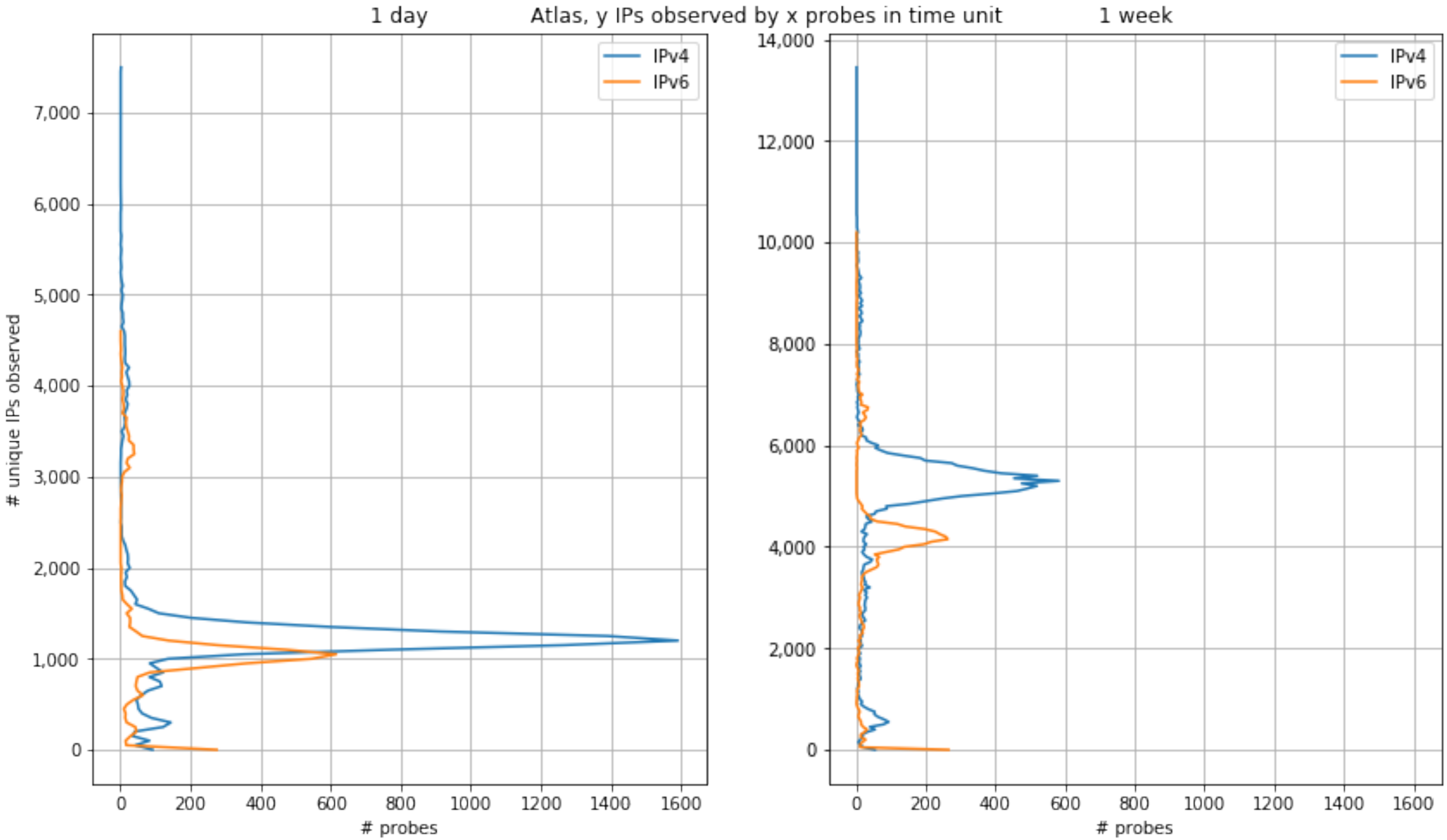
- start to look at the diversity in the collected IPv4 and IPv6 topology data
- both platforms

Atlas responding addrs



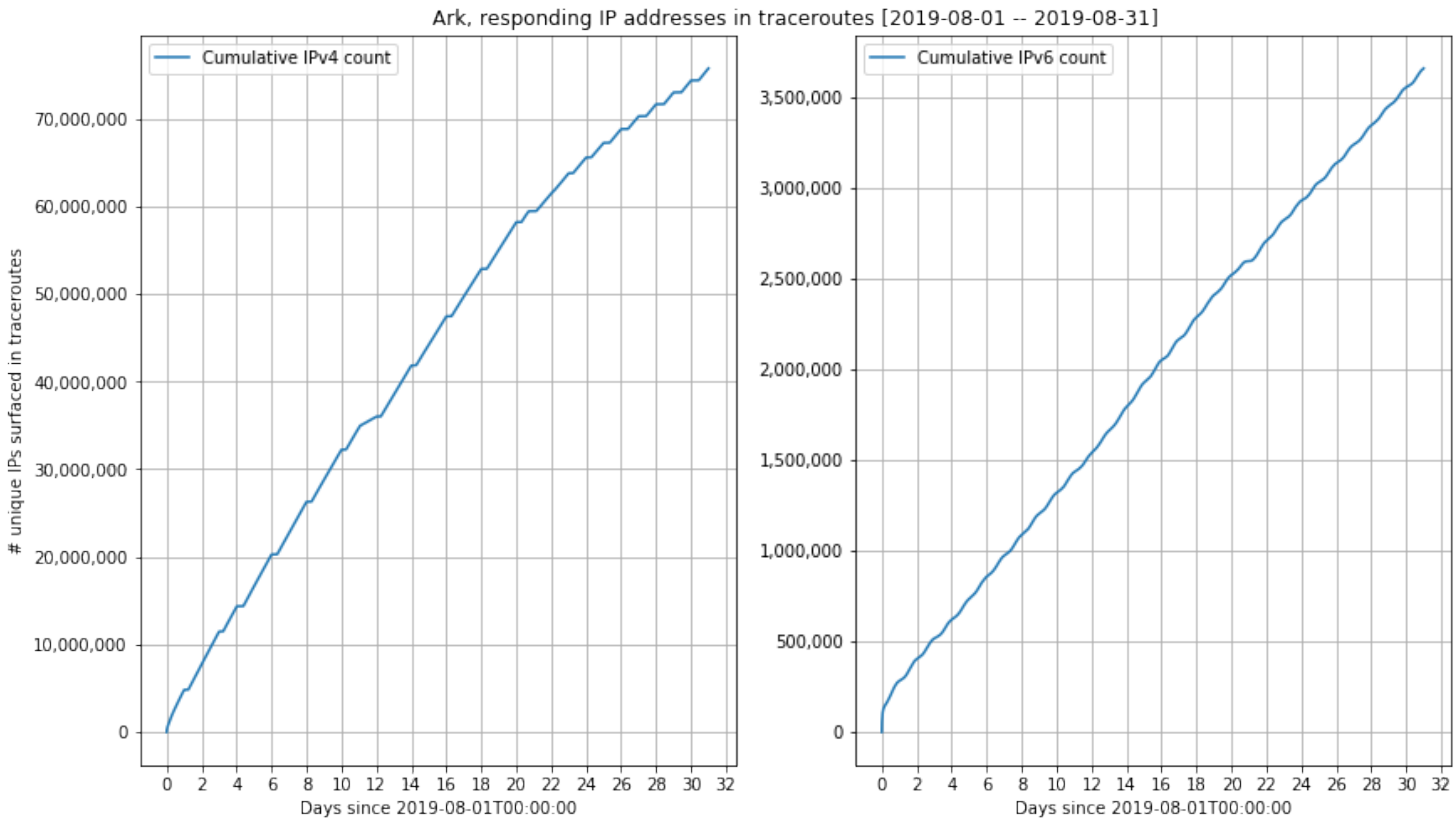
$14.40

Atlas responding addrs



$1.93

Ark responding addrs



$0.85

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

```
-- Select set of IPs here
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.sds_test.rib_20190424_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_lpm 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_lpm
  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:

IPv4	Atlas	Ark	Atlas ∩ Ark
Addrs	793,342	4,811,733	376,098
/24s	394,368	2,344,294	233,108
ASNs	45,154	36,589	31,033

IPv6	Atlas	Ark	Atlas ∩ Ark
Addrs	164,989	282,238	109,496
/48s	30,787	56,501	24,977
ASNs	8,305	10,165	7,757

$2.15

Observed per week:

IPv4	Atlas	Ark	Atlas ∩ Ark
Addrs	1,117,248	41,824,271	593,896
/24s	496,282	3,632,743	372,278
ASNs	45,160	36,589	31,037

IPv6	Atlas	Ark	Atlas ∩ Ark
Addrs	239,417	1,792,204	144,493
/24s	34,836	143,445	27,215
ASNs	8,307	10,165	7,757

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

ipFrom	dstAddress	dstName	startTime	af	parisId	msmId	prbId	hops.hop	hops.resultHops.err	hops.resultHops.rtt	hops.resultHops.from
2001:67c:6ec:201:145:220:0:55	2001:dc4::1	2001:dc4::1	2019-08-01 00:00:26 UTC	6	0	1430809	6031	1		0.7860000133514404	2001:67c:6ec:201:145:220:0:1
										0.8059999942779541	2001:67c:6ec:201:145:220:0:1
										0.5960000157356262	2001:67c:6ec:201:145:220:0:1
								2		0.7689999938011169	2001:7f8:1::a500:6939:1
										0.9020000100135803	2001:7f8:1::a500:6939:1
										0.7400000095367432	2001:7f8:1::a500:6939:1
								3		9.303999900817871	2001:470:0:431::2
										9.35099983215332	2001:470:0:431::2
										9.307000160217285	2001:470:0:431::2
								4		13.701000213623047	2001:470:0:410::2
										13.619999885559082	2001:470:0:410::2
								5		73.25399780273438	2001:470:0:440::1
										73.4219970703125	2001:470:0:440::1
										73.26699829101562	2001:470:0:440::1
								6		73.39399719238281	2001:470:0:20a::1
										73.28099822998047	2001:470:0:20a::1
										73.39900207519531	2001:470:0:20a::1
								7		95.32099914550781	2001:470:0:270::2
										92.66999816894531	2001:470:0:270::2
										95.2229995727539	2001:470:0:270::2
								8		103.51699829101562	2001:470:0:18e::2

```
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*

45	2019-07-12 01:00:00 UTC	4	192.168.1.10	188.176.98.1	83.88.1.81	
				83.88.1.81	198.32.118.22	
				198.32.118.22	209.148.237.1	
				209.148.237.1	209.148.229.225	
				209.148.229.225	209.148.235.234	
				209.148.235.234	209.148.227.138	
				209.148.227.138	209.148.233.98	
46	2019-07-12 01:00:00 UTC	4	192.168.1.10	188.176.98.1	83.88.1.81	
				83.88.1.81	198.32.118.22	
				198.32.118.22	209.148.231.49	
47	2019-07-12 01:00:00 UTC	4	192.168.1.10	188.176.98.1	83.88.1.81	
				83.88.1.81	198.32.118.22	
				198.32.118.22	209.148.231.49	
				209.148.231.49	209.148.229.225	
				209.148.229.225	209.148.235.234	
				209.148.235.234	64.71.240.2	
				64.71.240.2	69.63.243.42	

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)

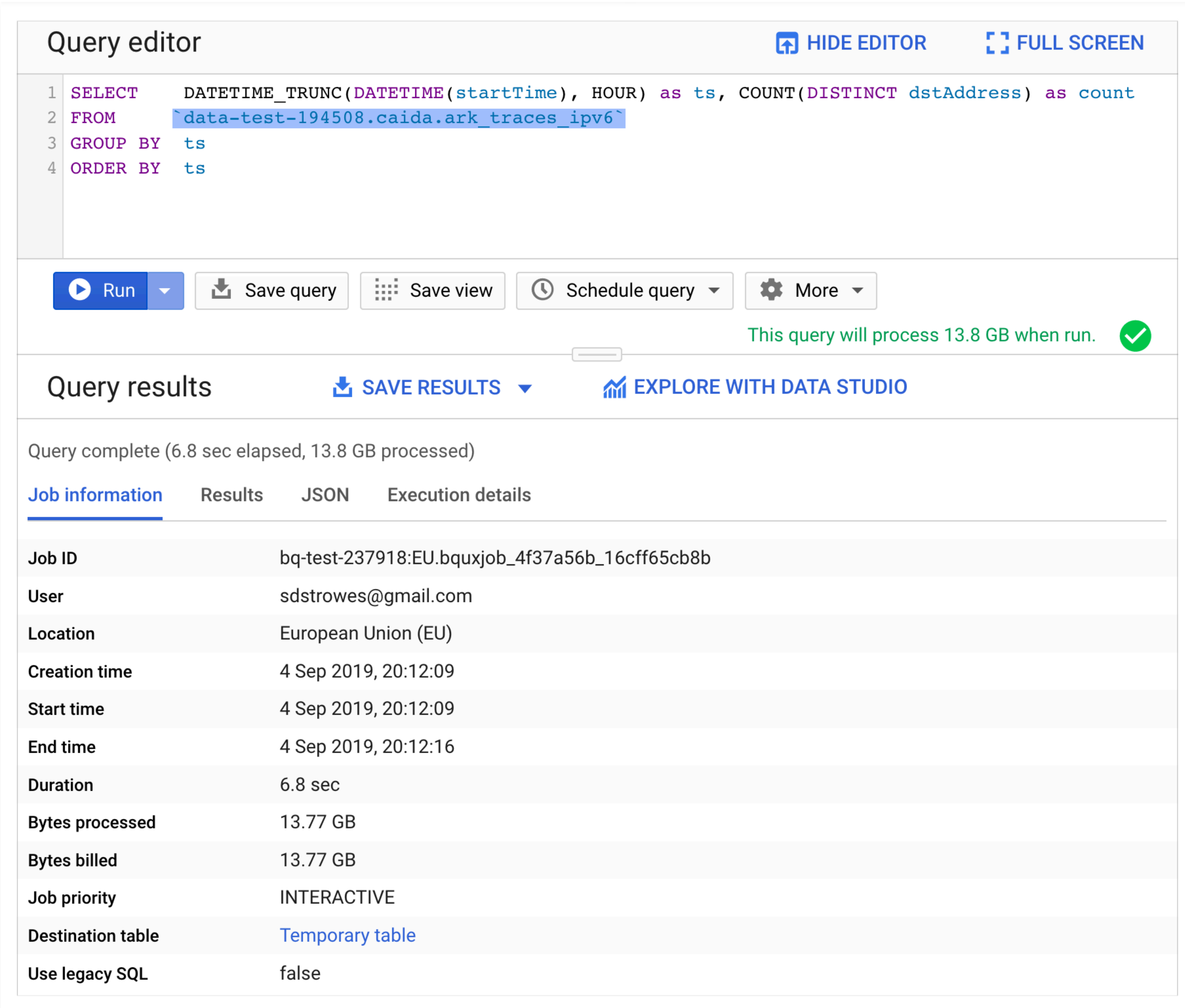
back to money

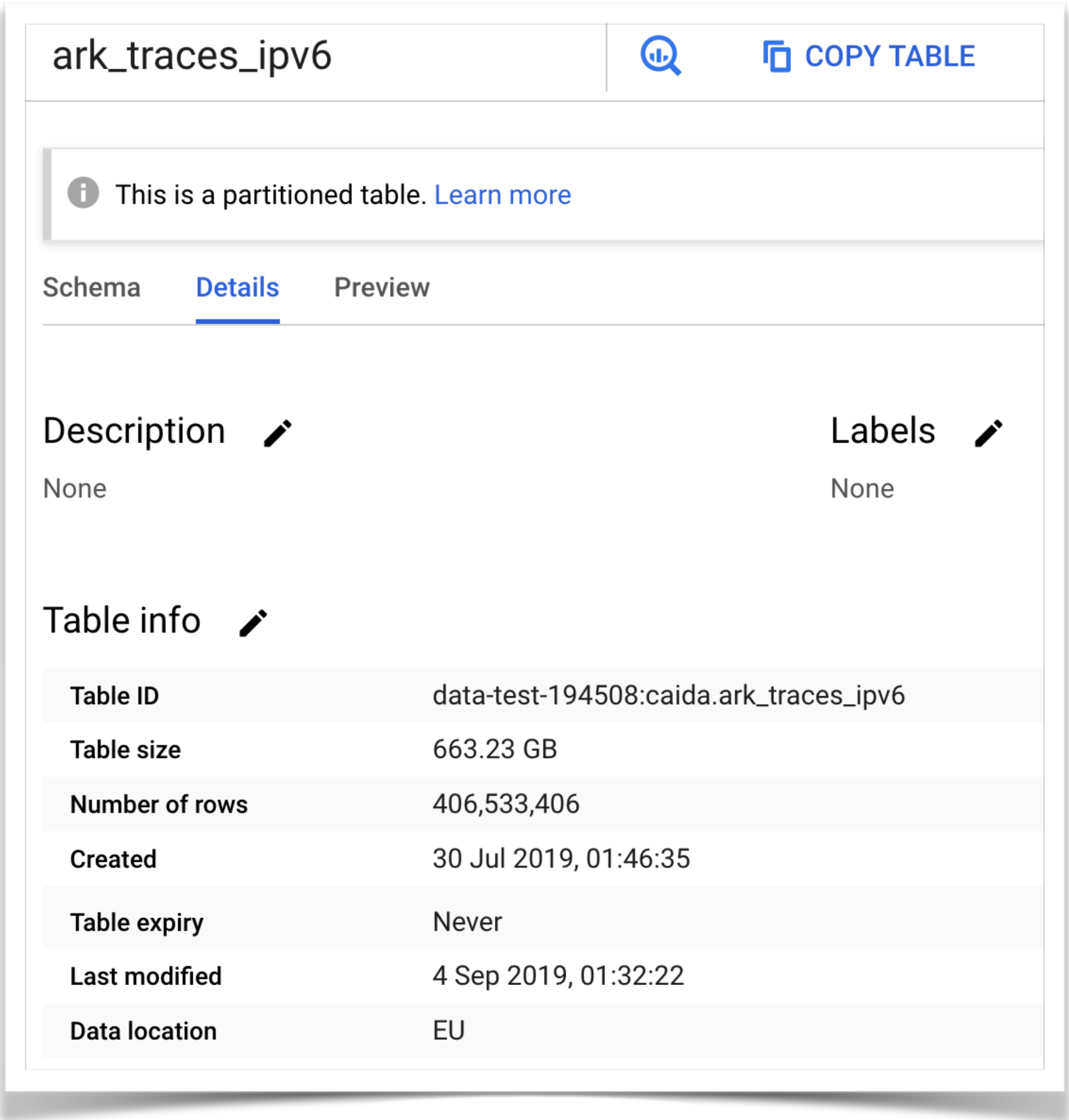
1–31 Aug 2019	BigQuery Analysis: 86.647 Tebibytes (Source:bq-test [bq-test-237918])	US$428.24

- 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://cloud.google.com/free/


Query editor
HIDE EDITOR
FULL SCREEN
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
Run
Save query
Save view
Schedule query
More
This query will process 13.8 GB when run.
Query results
SAVE RESULTS
EXPLORE WITH DATA STUDIO
Query complete (6.8 sec elapsed, 13.8 GB processed)
Job information
Results
JSON
Execution details
Job ID
bq-test-237918:EU.bquxjob_4f37a56b_16cff65cb8b
User
sdstrowes@gmail.com
Location
European Union (EU)
Creation time
4 Sep 2019, 20:12:09
Start time
4 Sep 2019, 20:12:09
End time
4 Sep 2019, 20:12:16
Duration
6.8 sec
Bytes processed
13.77 GB
Bytes billed
13.77 GB
Job priority
INTERACTIVE
Destination table
Temporary table
Use legacy SQL
false


ark_traces_ipv6
COPY TABLE
This is a partitioned table. Learn more
Schema
Details
Preview
Description
None
Labels
None
Table info
Table ID data-test-194508:caida.ark_traces_ipv6
Table size 663.23 GB
Number of rows 406,533,406
Created 30 Jul 2019, 01:46:35
Table expiry Never
Last modified 4 Sep 2019, 01:32:22
Data location EU

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