RIPE NCC

RIPE NETWORK COORDINATION CENTRE

Comparing RIPE Atlas and CAIDA Ark Data using BigQuery

Stephen Strowes <sds@ripe.net> | MAT WG, RIPE 79 | 2019-10-17

RIPE Atlas

- Collect a lot of measurement data
- We provide no general means to filter or analyse that data

CAIDA Ark

- https://www.caida.org/projects/ark/
- Goal: topology discovery

Google BigQuery

- General platform for data warehousing and analysis
- Feedback loop: “is this useful?”

Google BigQuery

- Can manage this much data without flinching
- Few hardware limitations
- Queries are fast, but not real-time
  - think: O(seconds — minutes), not O(milliseconds)
  - responsive enough to iterate during data exploration
- Allows exploration of the full dataset in new ways
  - ask for results related to ANY measurement
  - and you can combine with any other dataset you have access to

Using BigQuery: Ark + Atlas

- Two quick views of the data in this talk
  - How long it takes each dataset to reveal responsive ASNs
  - How long it takes each dataset to reveal AS adjacencies

Prefix Matching

```
SELECT    startTime, IpFrom, dstAddress, prbId, hops
FROM      `data-test-194508.prod.traceroute_atlas_prod`, UNNEST(hops) AS h, UNNEST(resultHops) AS rh
WHERE     af = 6
AND       (startTime >= "2019-08-01T00:00:00" AND startTime < "2019-08-08T00:00:00")
AND       NET.IP_FROM_STRING(rh.from)
  BETWEEN NET.IP_FROM_STRING("2001:db8::")
  AND     NET.IP_FROM_STRING("2001:db8:ffff:ffff:ffff:ffff:ffff:ffff")
```

Prefix Matching

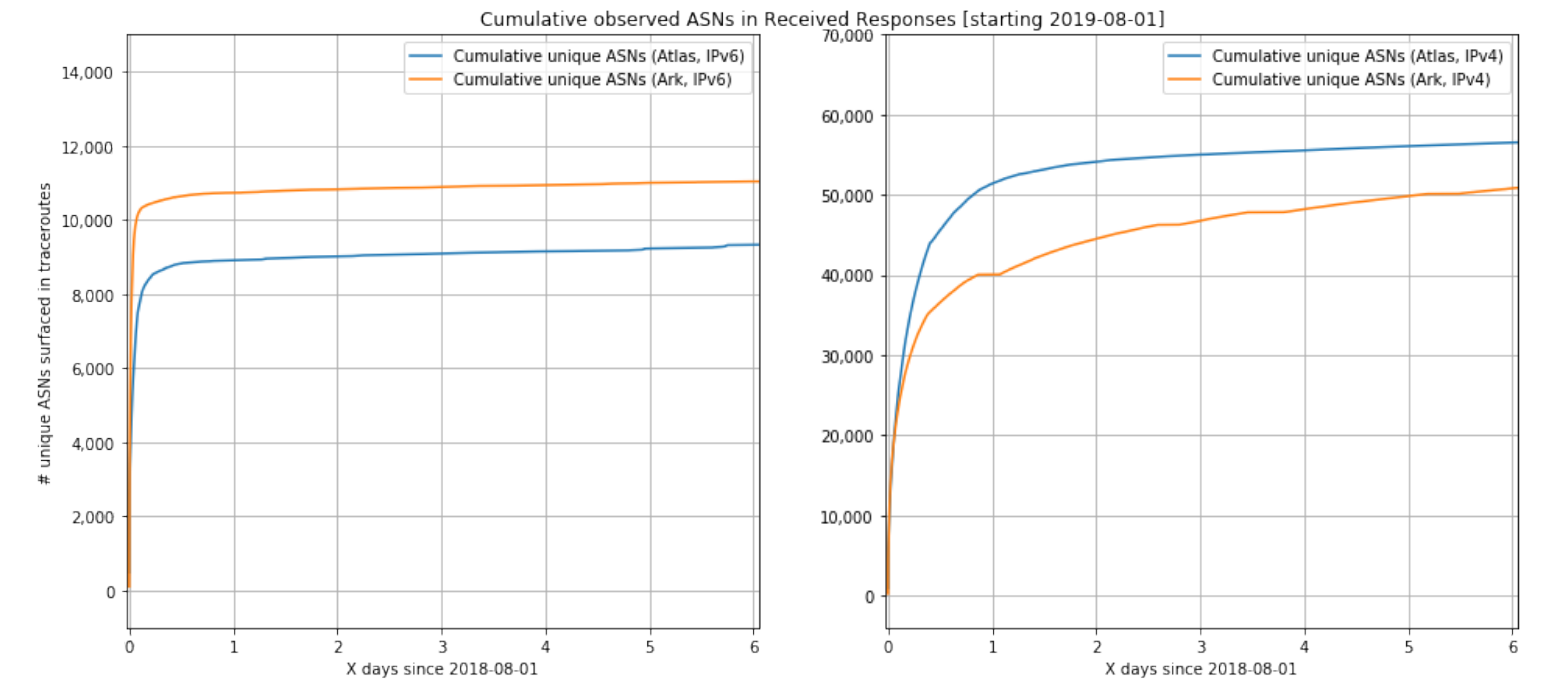
startTime	IpFrom	dstAddress	prbId	hop	rtt	from
2019-08-01 00:00:03 UTC	2001:db8:0:20:eade:27ff:fec9:71f0	2600:3c01::f03c:91ff:fec8:52fe	25054	1	0.4659999907016754	2001:db8:0:20::2
2019-08-01 00:00:03 UTC	2001:db8:0:20:eade:27ff:fec9:71f0	2600:3c01::f03c:91ff:fec8:52fe	25054	1	0.4320000112056732	2001:db8:0:20::2
2019-08-01 00:00:03 UTC	2001:db8:0:20:eade:27ff:fec9:71f0	2600:3c01::f03c:91ff:fec8:52fe	25054	1	0.42899999022483826	2001:db8:0:20::2
2019-08-01 00:00:05 UTC	2001:db8:0:20:eade:27ff:fec9:71f0	2600:3c03::f03c:91ff:fee4:7441	25054	1	0.4359999895095825	2001:db8:0:20::2
2019-08-01 00:00:05 UTC	2001:db8:0:20:eade:27ff:fec9:71f0	2600:3c03::f03c:91ff:fee4:7441	25054	1	0.4300000071525574	2001:db8:0:20::2
2019-08-01 00:00:05 UTC	2001:db8:0:20:eade:27ff:fec9:71f0	2600:3c03::f03c:91ff:fee4:7441	25054	1	0.46000000834465027	2001:db8:0:20::2
2019-08-01 00:00:09 UTC	2001:db8:4447:0:eade:27ff:fec9:7134	2a02:7d00:1:200:624:0:bc5c:29fa	26725	3	5.738999843597412	2001:db8:1101::23
2019-08-01 00:00:09 UTC	2001:db8:4447:0:eade:27ff:fec9:7134	2a02:7d00:1:200:624:0:bc5c:29fa	26725	3	5.760000228881836	2001:db8:1101::23
2019-08-01 00:00:09 UTC	2001:db8:4447:0:eade:27ff:fec9:7134	2a02:7d00:1:200:624:0:bc5c:29fa	26725	1	5.6479997634887695	2001:db8:4447::1

Rows per page: 100 1 - 100 of 659479 First page

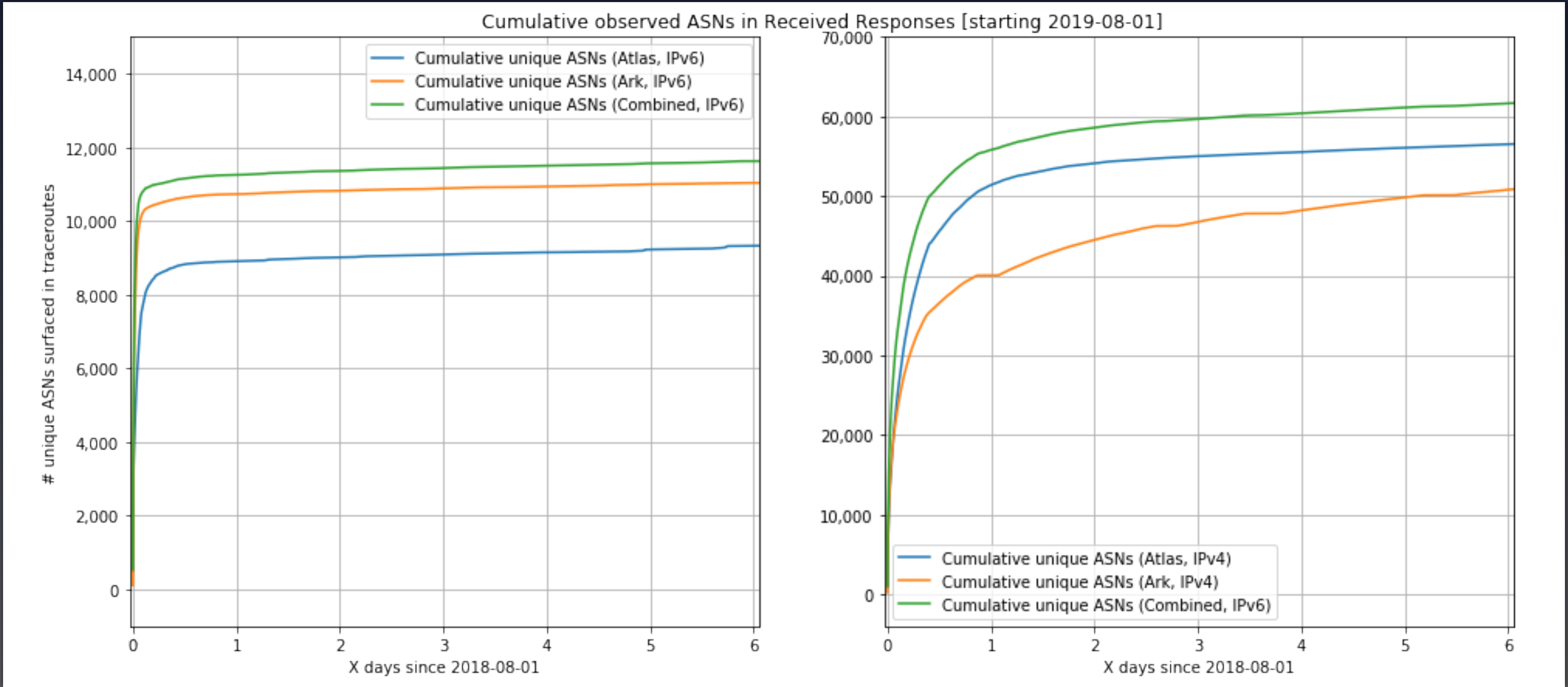
Longest Prefix Matching

```sql
WITH src_addrs AS
(
  SELECT DISTINCT srcAddress as ip
  FROM `data-test-194508.caida.ark_traces_ipv4`
  WHERE startTime >= @start_date AND startTime < @end_date AND srcAddress != ""
),
hop_addrs AS
(
  SELECT DISTINCT hopAddress as ip
  FROM `data-test-194508.caida.ark_traces_ipv4`, unnest(hops)
  WHERE startTime >= @start_date AND startTime < @end_date AND hopAddress != ""
),
ips AS
(
  SELECT DISTINCT ip FROM src_addrs
  UNION ALL
  SELECT DISTINCT ip FROM hop_addrs
),

bgp AS
(
  SELECT distinct prefix, network, start, foo.end, masklen, origin
  FROM  `sds_test.rrc0_rib` foo
  WHERE date = DATE(@start_date)
  AND af = 4
),

matched_v4_results AS
(
  SELECT ip, bgp.*
  FROM (
    SELECT *, NET.SAFE_IP_FROM_STRING(ip) & NET.IP_NET_MASK(4, masklen) start
    FROM ips, UNNEST(GENERATE_ARRAY(0,32)) masklen
    WHERE BYTE_LENGTH(NET.SAFE_IP_FROM_STRING(ip)) = 4
  )
  JOIN bgp
  USING (start, masklen)
),

matched_v4_results_lpm AS
(
  with a AS (
    select distinct ip, max(masklen) as masklen
    from matched_v4_results
    group by ip
  )
  SELECT distinct b.*
  FROM matched_v4_results b
  JOIN a
  USING (ip, masklen)
),

mix_in_missing AS
(
  SELECT ip, network, masklen, origin
  FROM   matched_v4_results_lpm

  UNION ALL

  SELECT *, null, null, null
  FROM ips
  WHERE ip not in (SELECT ip FROM matched_v4_results)
),

aggred_v4_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 mix_in_missing
  group by ip
)

SELECT   DATE(@start_date) as day, ip, origins
FROM     aggred_v4_asns
```

Responsive ASNs



Responsive ASNs, combined



Looking at Adjacencies

- First: build tables of IP_n — $IP_{(n+1)}$ adjacencies
- Then: build tables of AS_n — $AN_{(n+1)}$ adjacencies

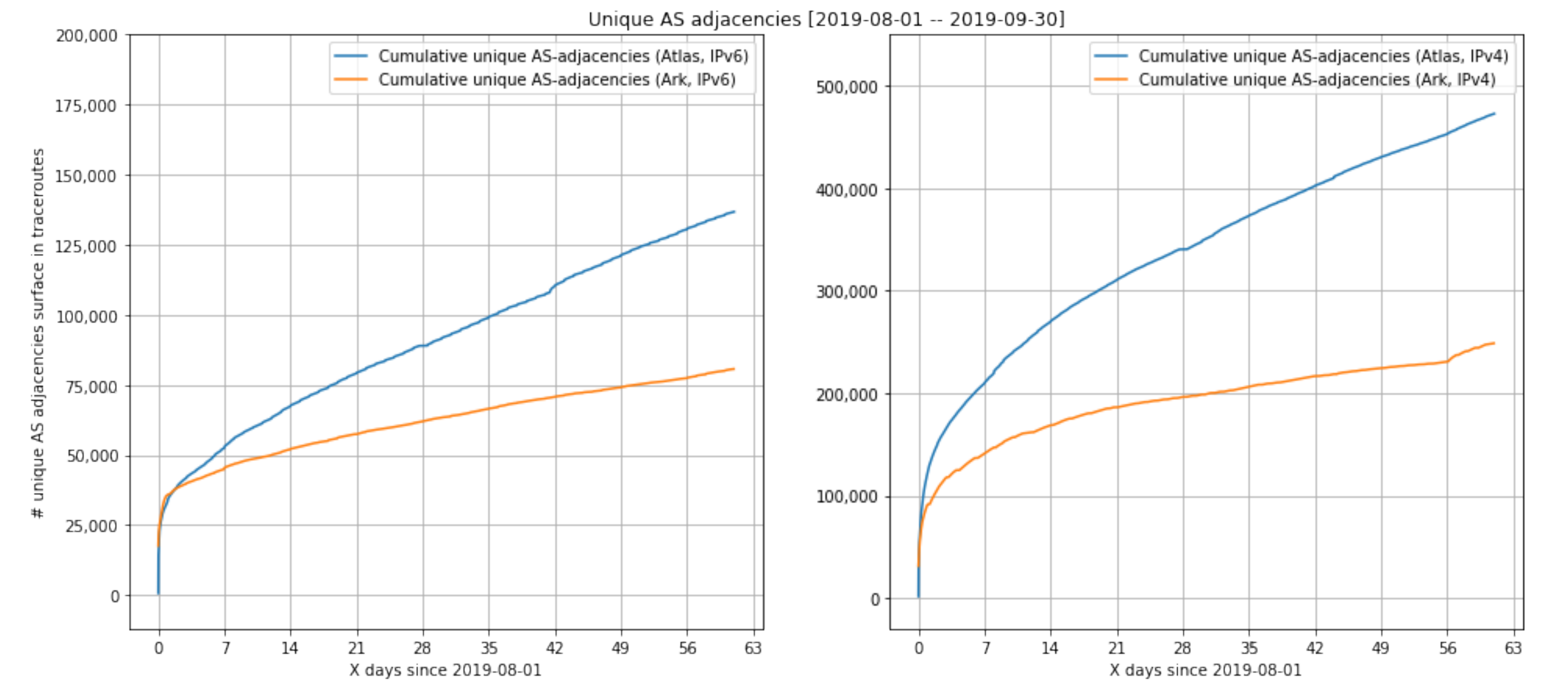
Looking at Adjacencies

- This is one place where SQL doesn't quite cut it
- But, BigQuery permits arbitrary code

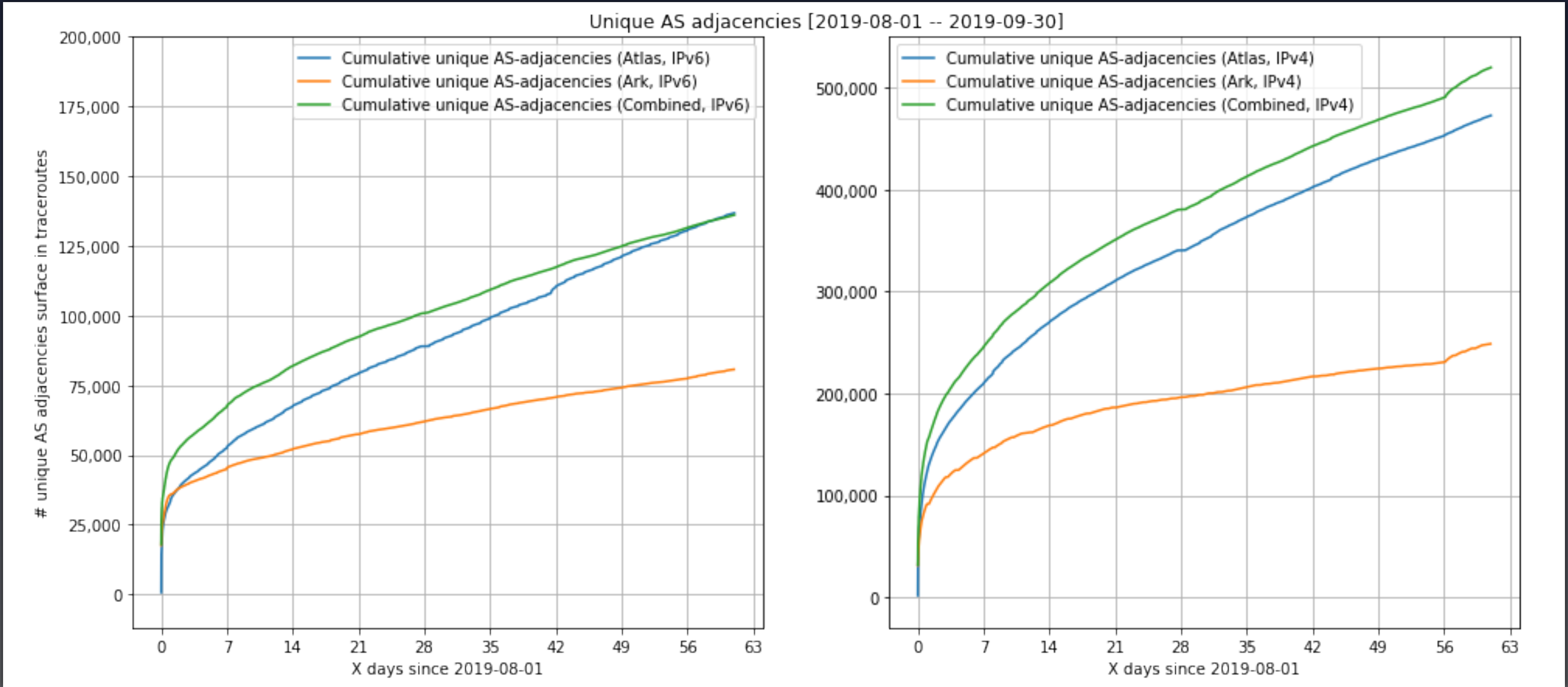
Arbitrary Code

```
CREATE TEMP FUNCTION GetIPs(json_str STRING)
RETURNS ARRAY<
  STRING
>
LANGUAGE js
AS """
  var row = JSON.parse(json_str);

  ips = new Set();
  for (var i = 0; i < row.length; i++) {
    this_hop = row[i]['resultHops'];
    this_hop.forEach(function(response, index) {
        ips.add(response['from'])
    });
  };

  return Array.from(ips);
""";

SELECT prbId, GetIPs(to_json_string(hops))
FROM `data-test-194508.prod.traceroute_atlas_prod`
WHERE startTime > "2019-08-01T00:00:00"
AND   startTime < "2019-08-02T00:00:00"
AND   af = 6
```

AS Adjacencies



AS Adjacencies, combined



Direction of Travel

- BigQuery doesn't solve all problems but it's extremely powerful
- Ultimately: open up access to (some of?) our data via BigQuery
- We'd like something that looks like the following:
  - **Storage cost, data availability**: the NCC's responsibility
  - **Computation cost and further storage**: on anybody who wants to query the data

Upcoming

- Articles on:
  - the work required in the BigQuery platform to query this data
  - the distinctions between the two datasets

Questions ?

Stephen Strowes <sds@ripe.net>
@sdstrowes