Popeye

Flight Tracking with SQLite

Keeping track of flights

- FlightAware needs a fast way to keep track of flights actually in the air.
- FlightAware receives input from thousands of sources, and generates a consistent stream of flight events.
    - This involves sorcery that will not be addressed here.
- These have to be turned into a view of what's currently in the air.
- This view needs to be up to date and super fast to query.

Birdseye

- For most of history this has been kept in speedtables.
  - This is Birdseye.
- This has worked pretty well, but has some problems.
  - There's more flights all the time, and reading and interpreting the stream of flight data in Tcl is getting tough.
  - Speedtables are volatile, so frequent snapshots need to be taken, and starting or restarting a Birdseye server is kind of slow.

Controlstream

- Flight events are transmitted, stored, and archived in a format called "daystream".
    - https://www.tcl.tk/community/tcl2017/assets/talk95/Paper.pdf
- Each flight event (position, arrival, departure, etc...) is a separate line.
- Each line are tab-separated key-value pairs.
- First two pairs are a timestamp composed of seconds and a sequence number.

Controlstream

- Example:

```
_c	1539354574	_s	10	type	position	ident	WZZ1022	childID	126	_t	b
adhoc	1	adsb_category	A3	adsb_version	2	airground	A	alt	370
alt_ft	37000	alt_gnss	38250	alt_src	A	bitmask	0	clock	1539354569
	combid	1539354574-615	facility	fAT-42bdf278-a9ba-412b-8bbf-9452990c3965
	feed	ADEPT7	feed_c	1539354569	feed_s	2546	flightlevel	370	fp
WZZ1022-1539302978-ed-0003:5	gSource	feed	gs	507	hSource	feed	heading	130
heading_magnetic	127.6	hexid	471F61	lat	55.91468	lon	12.35473	mach	0.768
	nac_p	[...]
```

- Zillions of these a second.

Controlstream

- The daystream feed containing the canonical view of flight events is called "controlstream".
- This is the "input" side of Birdseye

Trackstream

- Clients, like webservers, query birdseye using a protocol called "trackstream".
  - It's your basic query-response TCP/IP API, send a query and get a one-line (maybe a very long line) reply.
  - Queries actually Tcl and look kind of like like Speedtable search queries.
    - Trackstream was the inspiration for speedtables.
    - Not quite the same, speedtable syntax is more database-like so Birdseye translates queries.
  - Replies are almost always Tcl lists, except where they're tab-separated lists for some historic reason.

Eagle Eye

- Build one to throw away
- Eagle-Eye replaced Speedtables with a Cassandra cluster
- Instead of having a bunch of Birdseye servers, and trackstream queries, clients would connect to Cassandra and make CQL queries.
- Massively multithreaded controlstream reader to populate Cassandra.

Eagle Eye Problems

- CQL is not as powerful as Speedtables API
  - This was actually a surprise, it's less powerful than SQL, sure, but Speedtables is not even pretending to be SQL.
- Massive write multiplication. You need a separate copy of a table for each "index".
- Cassandra latency meant keeping a lot of duplicated state while reading controlstream, which meant startup delays as this state was restored.
- Getting good performance for some queries required making many CQL queries in parallel, and changes to the webservers.
- Basically, Cassandra is not a great tool for a general query engine.

Popeye

- Popeye replaced Speedtables with SQLite
- SQLite is non-volatile, so a popeye can be shut down and started up without delay.
- SQL is far richer than Speedtables API (STAPI) or CQL
- SQLite latency is low because there's no network I/O
    - Still higher than speedtables, but not enough to be a killer.

Popeye - Input

- Popeye replaced Tcl controlstream reader with one in C++
- C++ is lower level than Tcl, but C++14 is pretty good.
- Plus, we have revived a nice C++ Tcl wrapper library.
  - https://github.com/flightaware/cpptcl
  - Shannon's talk will cover this.

Popeye - Input

- Can't practically query SQLite for keeping track of state for every flight data event, so some processing was simplified or deferred to the back end.
  - Maintaining the bounding box is handled in a C++ map.
  - Projected positions are just stored and filtered in trackstream.
- Some state needs to be maintained, but occasional SQLite queries and caching results in maps is good enough.
  - So no massive state restore at startup.

Popeye - Input

- Generating SQL code for updating flight status was actually taking significant CPU time in C++ string operations.
  - We walk the column list and build a tree, and only generate new SQL when a new leaf node is created.
  - The tree is brute force and never pruned because it tops out at 1500-2000 unique statements.

Popeye - Output

- We already had experience converting Speedtables queries to SQL
  - "STAPI" - Speedtables API: maps queries to trackstream-style sockets, or PostgreSQL database
- Doesn't actually use STAPI, because trackstream is not exactly speedtables, just similar.
- Birdseye code was already translating trackstream to speedtables, this was modified to generate SQL.

Popeye - Trackstream

- Example query

```
search -inAir both -originOrDestination KTPA -withPositionsSince 999999 -unblock
""
```

- Search flights, arriving or departing KTPA, with their track. But don't show blocked (hidden) flights.

Popeye - Trackstream

- This generates some SQL

```
SELECT fp,[...],inflight.clock as clock,inflight.ident as ident FROM inflight LEFT
JOIN blocked on inflight.ident = blocked.ident WHERE (orig = 'KTPA' OR dest =
'KTPA') AND inflight.clock > '1539346626' AND lat IS NOT NULL AND blocked.clock IS
NULL ORDER BY clock DESC LIMIT 12800;
```

- Then popeye runs over the resulting flights and builds a track for each aircraft:

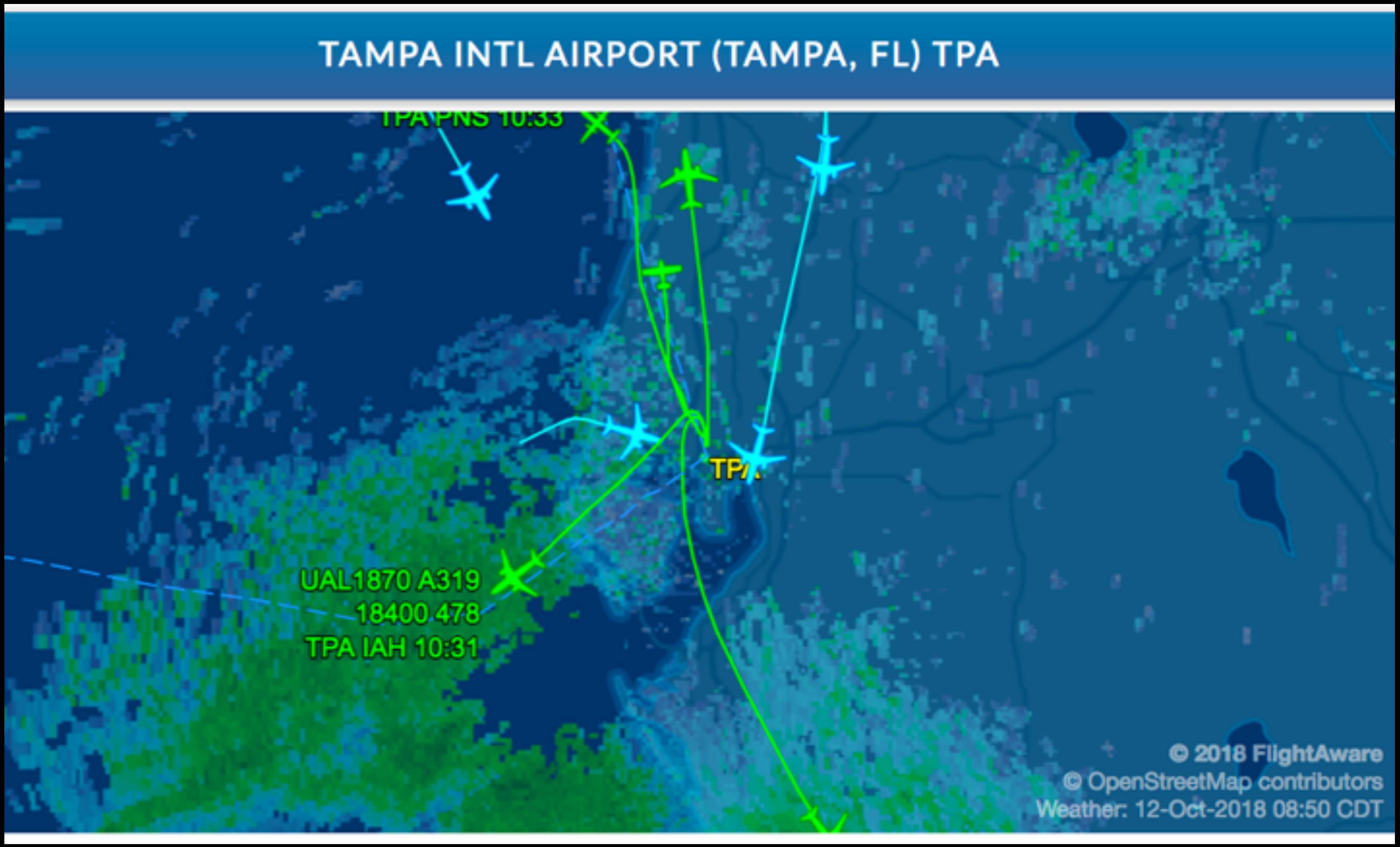
```
SELECT lon, lat FROM positions WHERE fp = :fp AND clock > :withPositionsSince AND
(gs <> 0 OR gs IS NULL) ORDER BY clock;
```

Popeye - Trackstream

- And that produces a Tcl list


```
{ident SWA104 prefix {} type B737 suffix {} origin KDCA destination KTPA
departureTime 1539351420 faFlightID SWA104-1539149199-airline-0115 waypoints
{38.85 -77.04 [...] 27.98 -82.53} blocked 0 timeout ok timestamp 1539351989
firstPositionTime 1539351463 lowLatitude 38.57555 lowLongitude -77.46361
highLatitude 38.98889 highLongitude -77.04103 longitude -77.46361 latitude
38.57555 groundspeed 381 altitude 198 altitudeFeet 19800 altitudeStatus -
updateType TZ altitudeChange D heading 206 arrivalTime 0 estimatedArrivalTime
1539358800 track {-77.041 38.872 -77.113 38.922 -77.164 38.973 -77.256 38.989
-77.35 38.977 -77.359 38.872 -77.344 38.764 -77.403 38.67 -77.464 38.576}} [...]
```

Resulting Webpage



Popeye - Trackstream

- Then when you look at a particular flight:

```
info UAL735-1539101593-fa-0010
```

- Which uses a somewhat simpler query:

```
select * from inflight where fp = :fp order by clock desc limit 1;
```

- And produces:

```
ident UAL735 prefix {} type B738 suffix {} origin KTPA destination KIAD departureTime
1539352740 faFlightID UAL735-1539101593-fa-0010 waypoints {27.98 -82.53 [...] 38.95
-77.46} blocked 0 timeout ok timestamp 1539353470 firstPositionTime 1539352818 lowLatitude
28.00000 lowLongitude -82.64333 highLatitude 29.06889 highLongitude -82.52472 longitude
-82.64333 latitude 29.06889 groundspeed 436 altitude 235 altitudeStatus {} altitudeFeet
23500 updateType TZ altitudeChange C heading 353 estimatedArrivalTime 1539359820
```

Popeye - Trackstream

- And for the track:

```
get_track UAL735-1539101593-fa-0010
```

- This hits the position history table:

```
SELECT clock, lon, lat, gs, alt, alt_ft, altChar, updateType, coalesce(cid,'---') cid, facility, altChange,
heading FROM positions WHERE fp = :fp ORDER BY clock;
```

- And produces:

{1539352818 -82.53333 28.00000 169 8.00 {} TZ --- KTPA D {} 800} {1539352911 -82.52472 28.08389 217 39.00 {} TZ --- KZJX C 5 3900} {1539352973 -82.52583 28.16639 275 69.00 {} TZ --- KZJX C 359 6900} {1539353035 -82.53806 28.24556 285 100.00 {} TZ --- KZJX C 352 10000} {1539353097 -82.55111 28.35778 322 115.00 {} TZ --- KZJX C 354 11500} {1539353159 -82.56167 28.46500 361 138.00 {} TZ --- KZJX C 355 13800} {1539353222 -82.57750 28.58417 398 160.00 {} TZ --- KZJX C 353 16000} {1539353284 -82.59194 28.69806 409 180.00 {} TZ --- KZJX C 354 18000} {1539353347 -82.61111 28.82444 422 201.00 {} TZ --- KZJX C 352 20100} {1539353408 -82.62695 28.94333 426 219.00 {} TZ --- KZJX C 353 21900} {1539353470 -82.64333 29.06889 436 235.00 {} TZ --- KZJX C 353 23500} {1539353532 -82.66000 29.19167 439 250.00 {} TZ --- KZJX C 353 25000}

Resulting Webpage

Popeye - Trackstream

- But that track information is kind of oldschool, so now we do:

```
get_track_kv UAL735-1539101593-fa-0010
```

- Which uses a somewhat simpler query:

```
SELECT clock,lon,lat,gs,alt_ft,[...],nav_qnh,emergency FROM positions WHERE fp = :fp ORDER
BY clock
```

- Which produces:

```
{lon -82.53333 gs 169 alt_ft 800 clock 1539352818 heading {} lat 28.00000 updateType TZ facility KTPA}
{lon -82.52472 gs 217 alt_ft 3900 clock 1539352911 heading 5 lat 28.08389 updateType TZ facility KZJX}
{lon -82.52583 gs 275 alt_ft 6900 clock 1539352973 heading 359 lat 28.16639 updateType TZ facility KZJX}
{lon -82.53806 gs 285 alt_ft 10000 clock 1539353035 heading 352 lat 28.24556 updateType TZ facility KZJX}
{lon -82.55111 gs 322 alt_ft 11500 clock 1539353097 heading 354 lat 28.35778 updateType TZ facility KZJX}
{lon -82.56167 gs 361 alt_ft 13800 clock 1539353159 heading 355 lat 28.46500 updateType TZ facility KZJX}
```

Popeye - Trackstream

- That looks just like a nicer format, but it allows us to include more information in the result. Like this:


```
{lon -11.24672 gs 506 alt_ft 36000 pos_nic 8 alt_gnss 34800 clock 1539354680
speed_ias 282 heading 294 mach 0.844 lat 58.32431 vertRate_geom -64 vertRate 128
updateType TA facility fAT-52c781b2-d33b-40a5-b2b8-94051620eb4b pos_rc 186
heading_magnetic 293 adsb_version 0}
{lon -11.36881 gs 504 alt_ft 35975 pos_nic 8 alt_gnss 34775 clock 1539354710
heading 294 lat 58.35329 vertRate_geom 0 updateType TA facility fAT-52c781b2-
d33b-40a5-b2b8-94051620eb4b pos_rc 186 adsb_version 0}
{speed_tas 498 lon -11.52422 gs 504 alt_ft 36000 pos_nic 8 alt_gnss 34775 clock
1539354748 nav_alt 36000 heading 294 roll 0.0 lat 58.38991 nav_qnh 1013.0
vertRate_geom 0 updateType TA facility fAT-52c781b2-d33b-40a5-b2b8-94051620eb4b
pos_rc 186 adsb_version 0}
{lon -11.64468 gs 505 alt_ft 35975 pos_nic 8 alt_gnss 34750 clock 1539354778
heading 294 lat 58.41817 vertRate_geom 0 updateType TA facility fAT-52c781b2-
d33b-40a5-b2b8-94051620eb4b pos_rc 186 adsb_version 0}
```

Popeye - Trackstream

- This is new "MODE S" information reported from the autopilot, and includes things like the desired altitude (nav_alt) and heading (nav_heading).
- The information on the previous slide was from a 777 en-route from London to Houston.
- This has just been put into production.
- It's much easier to make changes like this with a full SQL database in our pocket.