All the Ships in the World

Francesc Campoy, Kurt Schwehr and Mano Marks
Tons of ships!

75K Ships
400M Points
What we're doing

Broadcasts → Data Collection → Storage → Cloud Computing Resources → Users

AIS Ship Tracking Sources
use ALL the Clouds!!!
Gathering Data
What is AIS?

- VHF transceivers on ships
- Designed in the 1990's
- Help ships navigate
- Line of sight
<table>
<thead>
<tr>
<th>Stats</th>
<th>Ship Positions</th>
<th>Unique Vessels</th>
<th>Date Range for demo</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpaceQuest Satellite</td>
<td>40M</td>
<td>76K</td>
<td>May 2012-present</td>
</tr>
<tr>
<td>Global Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stats</td>
<td>Ship Positions</td>
<td>Unique Vessels</td>
<td>Date Range for demo</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>SpaceQuest Satellite Global Coverage</td>
<td>40M</td>
<td>76K</td>
<td>May 2012-present</td>
</tr>
<tr>
<td>NOAA, Boston, MA</td>
<td>364M</td>
<td>9K</td>
<td>2005-Present</td>
</tr>
<tr>
<td></td>
<td>Ship Positions</td>
<td>Unique Vessels</td>
<td>Date Range for demo</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>SpaceQuest Satellite</td>
<td>40M</td>
<td>76K</td>
<td>May 2012-present</td>
</tr>
<tr>
<td>Global Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOAA, Boston, MA</td>
<td>364M</td>
<td>9K</td>
<td>2005-Present</td>
</tr>
<tr>
<td>Google, San Francisco, CA</td>
<td>240K</td>
<td>171</td>
<td>Last week</td>
</tr>
<tr>
<td>Stats</td>
<td>Ship Positions</td>
<td>Unique Vessels</td>
<td>Date Range for demo</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>SpaceQuest Satellite</td>
<td>40M</td>
<td>76K</td>
<td>May 2012-present</td>
</tr>
<tr>
<td>Global Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOAA, Boston, MA</td>
<td>364M</td>
<td>9K</td>
<td>2005-Present</td>
</tr>
<tr>
<td>Google, San Francisco, CA</td>
<td>240K</td>
<td>171</td>
<td>Last week</td>
</tr>
<tr>
<td>Google, Sydney, Australia</td>
<td>134K</td>
<td>48</td>
<td>Last week</td>
</tr>
</tbody>
</table>
All Most of the Ships in the World
Challenges

- Data size
- Data quality
- Overlapping messages
Storing Data
Cloud Storage

- Access Control Lists (ACLs)
- Up-to and beyond 4 PB
- Fast access to data
Architecture: storing data

Data Loggers
gsutil and boto upload

Cloud Storage
gs://bucket/

?
Compute Engine

- Generic Linux instances
- Run code in any language
- Service accounts make security easy
Architecture: storing data

<table>
<thead>
<tr>
<th>MMSI</th>
<th>Navigation Status</th>
<th>ROT</th>
<th>SOG</th>
<th>Longitude</th>
<th>Latitude</th>
<th>COG</th>
<th>True Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>319353000</td>
<td>8</td>
<td>-17</td>
<td>10.2</td>
<td>-122.4005</td>
<td>37.81267</td>
<td>129</td>
<td>130</td>
</tr>
</tbody>
</table>

Open Source C++ decoder: [libais](https://github.com/libais)
Laptop $ gcutil listimages --format=names
proc-ais-2013-05-14

Laptop $ gcutil listinstances --format=names
proc-ais10
proc-ais11
proc-ais5
proc-ais7
proc-ais8
proc-ais9

Laptop $ gcutil addinstance --persistent_boot_disk=true proc-ais12 --image=proc-ais-2013-05-14 --machine_type=n1-standard-1 --zone=us-central1-b --service_account_scopes=bigquery,compute-rw,storage-full,taskqueue
Laptop $ gcutil addinstance --persistent_boot_disk=true proc-ais10 --image=proc-ais-2013-05-14 --machine_type=n1-standard-1 --zone=us-central1-b --service_account_scopes=bigquery,cloudsql,compute-rw,storage-full,taskqueue
INFO: Waiting for insert of disk boot-proc-ais10. Sleeping for 3s.

Laptop $ gcutil ssh proc-ais10
INFO: Zone for 'proc-ais10' detected as u'us-central1-b'.
schwehr@proc-ais10:~$
Architecture: Bucket Notifications & Task Queues

Data Loggers
  gsutil and boto upload

Cloud Storage
  gs://bucket/

Convert to CSV
Serving data
Big Query

- Query terabyte data on a scale of seconds, not minutes (or hours)
- Write queries fast with a SQL-like syntax
- RESTful API means easy integration
Loading data to BigQuery
Bucket notifications, App Engine, and BigQuery
### Table Details: pos123

#### Table Info

<table>
<thead>
<tr>
<th>Table ID</th>
<th>ais-demo-v1.io.pos123</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Size</td>
<td>69.2 GB</td>
</tr>
<tr>
<td>Number of Rows</td>
<td>374,601,235</td>
</tr>
<tr>
<td>Creation Time</td>
<td>7:43pm, 22 Apr 2013</td>
</tr>
<tr>
<td>Last Modified</td>
<td>4:11pm, 26 Apr 2013</td>
</tr>
</tbody>
</table>

#### Preview

<table>
<thead>
<tr>
<th>Row</th>
<th>id</th>
<th>repeat_indicator</th>
<th>mmsi</th>
<th>nav_status</th>
<th>rot_over_range</th>
<th>rot</th>
<th>sog</th>
<th>position_accuracy</th>
<th>x</th>
<th>42.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>366890370</td>
<td>0</td>
<td>false</td>
<td>0.0</td>
<td>0.10000000149</td>
<td>0</td>
<td>-71.0256930786</td>
<td>42.3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>367010390</td>
<td>0</td>
<td>true</td>
<td>-731.386474609</td>
<td>7.800000019073</td>
<td>0</td>
<td>-70.9810595703</td>
<td>42.3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>367010390</td>
<td>0</td>
<td>true</td>
<td>-731.386474609</td>
<td>7.800000019073</td>
<td>0</td>
<td>-70.9810595703</td>
<td>42.3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>636090339</td>
<td>5</td>
<td>false</td>
<td>0.0</td>
<td>0.10000000149</td>
<td>0</td>
<td>-70.8772583008</td>
<td>42.5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>667029470</td>
<td>0</td>
<td>true</td>
<td>-731.386474609</td>
<td>0.20000000298</td>
<td>0</td>
<td>-71.0401000977</td>
<td>42.3</td>
</tr>
</tbody>
</table>
How fast is fast?

```
SELECT id, lon, lat, t
FROM (
    SELECT mmsi AS id,
    HOUR(rcvr_time) as hour,
    AVG(x) as lon,
    AVG(y) as lat,
    INTEGER(AVG(TIMESTAMP_TO_SEC(rcvr_time))) as t
    FROM [io.pos123]
    WHERE DATEDIFF(CURRENT_TIMESTAMP(), rcvr_time) < 90
    AND x <= 180 AND y <= 90
    AND mmsi >= 100000000 AND mmsi <= 999999999
    GROUP BY id, hour
    ORDER BY id, hour
) 
```
SELECT id, lon, lat, t
FROM {
  SELECT
    mmsi AS id,
    HOUR(rcvr_time) as hour,
    AVG(x) as lon,
    AVG(y) as lat,
    INTEGER(AVG(TIMESTAMP_TO_SEC(rcvr_time))) as t
  FROM [io.pos123]
  WHERE DATEDIFF(CURRENT_TIMESTAMP(), rcvr_time) < 90
  AND x <= 180 AND y <= 90
  AND mmsi >= 100000000 AND mmsi <= 999999999
  GROUP BY id, hour
  ORDER BY id, hour
}

Query Results  9:55pm, 8 May 2013

<table>
<thead>
<tr>
<th>Row</th>
<th>id</th>
<th>lon</th>
<th>lat</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>210348000</td>
<td>-70.99343719482718</td>
<td>42.387852503854559</td>
<td>1361897990</td>
</tr>
<tr>
<td>2</td>
<td>210348000</td>
<td>-70.97638637388977</td>
<td>42.3462934121074</td>
<td>1361913635</td>
</tr>
<tr>
<td>3</td>
<td>210348000</td>
<td>-70.86484985352</td>
<td>42.38179375784</td>
<td>1361833531</td>
</tr>
<tr>
<td>4</td>
<td>210348000</td>
<td>-70.54536020755938</td>
<td>42.61340510844843</td>
<td>1361800768</td>
</tr>
<tr>
<td>5</td>
<td>210348000</td>
<td>-70.91886259104795</td>
<td>42.37115990625695</td>
<td>1361905501</td>
</tr>
</tbody>
</table>
Serving architectures

Using BigQuery REST API from JavaScript
Serving architectures

Using BigQuery REST API from App Engine with Go
Serving architectures

Materializing query results on datastore
Serving architectures

Caching the formatted results on memcache
Serving architectures

Speed! Speed! Speed!

![Bar chart showing request time for BigQuery, Datastore, and Memcache]
Go on App Engine

Go as the language for the cloud

- Speed!
  - Fastest instance startup time
  - Compiled to machine code
  - Less CPU, fewer instances

- Memory efficiency
  - Fewer, more affordable instances

- Concurrency
  - Concurrent I/O and network ops
  - Easy performance increase
i := &memcache.Item{
    Key:    key,
    Value:  value,
}
memcache.Set(c, i)
Go
Concurrency made easy

```go
func() {
    i := &memcache.Item{
        Key:   key,
        Value: value,
    }
    memcache.Set(c, i)
}()
```
Go
Concurrency made easy

```go
errc := make(chan error)
go func() {
    i := &memcache.Item{
        Key: key,
        Value: value,
    }
    errc <- memcache.Set(c, i)
}() <-errc
```
Go
Concurrency made easy

```go
errc := make(chan error, 1)
go func() {
    i := &memcache.Item{
        Key:   key,
        Value: value,
    }
    errc <- memcache.Set(c, i)
}()
select {
case <-errc:
case <-time.After(time.Second):
}
```
Talks today
2:35pm - Room 7 - High Performance Apps with Go on App Engine
4:25pm - Room 7 - Advanced Go Concurrency Patterns
5:20pm - Room 2 - Fireside Chat with the Go team

Codelab tomorrow
9:00am - Room 3 - Whispering Gophers: network programming in Go

Office hours
3:00-3:45pm today | 1:45-2:30pm tomorrow
Displaying data
The Google Maps API

- Global and local context
- Multiplatform
- CanvasLayer syncs WebGL with Maps in JS
Not only ships! Any kind of sensor data would do

- Flights
- Taxi/car service
- Trucks
- Anything else
Thank you!

+ManoMarks
+FrancescCampoyFlores
+KurtSchwehr