GLARE DETECTION ARCHITECTURE

Version 1.0

Chris Bush
chris@nacho.fm

TABLE OF CONTENTS

ARCHITECTURAL GUIDELINES

- All boundaries between services should have clearly defined API contracts.
- REST services should use OpenAPI v3 as their specification standard.
- All other API definitions (such as message queue messages, websocket message, gRPC, etc) should have clearly defined specifications published before delivery.

GLARE DETECTION SERVICE (SINGLE IMAGE)

Service Requirements

The Glare Detection Service is a RESTful service that is intended to take image metadata from automated vehicles and determine if the image from the viewpoint of the automobile contains glare from the sun.

The following data is used to make this determination:

- Latitude: a float between 0 to 90 that shows the latitude in which the image was taken
- Longitude: a float between -180 to 180 that shows the longitude in which the image was taken
- Epoch: Linux epoch in seconds
- Orientation: a float between -180 to 180 representing the eastward orientation of car travel from true north. i.e. 0 means north, 90 east and -90 west
- Solar Position: The position of the sun determined by the given latitude, longitude and epoch. Specifically, the NREL SPA algorithm is used to calculate the solar position. See https://midcdmz.nrel.gov/spa/ for more information on the algorithm.
- Cloud Cover: Historical weather given the epoch, latitude, and longitude can be retrieved by using the Weatherstack historical

API (https://api.weatherstack.com/historical). Specifically, the cloud cover field in the response of that API should be used to determine the possibility of glare. (TODO: Need to determine what values are considered “sunny”. Unlikely to be 0, glare would be possible still with very slight cloud cover).

Glare is determined possible if:

- Azimuthal difference sun and the direction of the car travel is less than 30 degrees
- Altitude of the sun is less than 45 degrees
- The weather is sunny (i.e. there is little to no cloud cover)

OpenAPI Specification

https://github.com/nacho-fm/glare/blob/main/glare/openapi/detect_glare.yaml

Service Architecture

(Note: The batch message queue consumer needed by the Image Metadata Collection Service is not shown here)

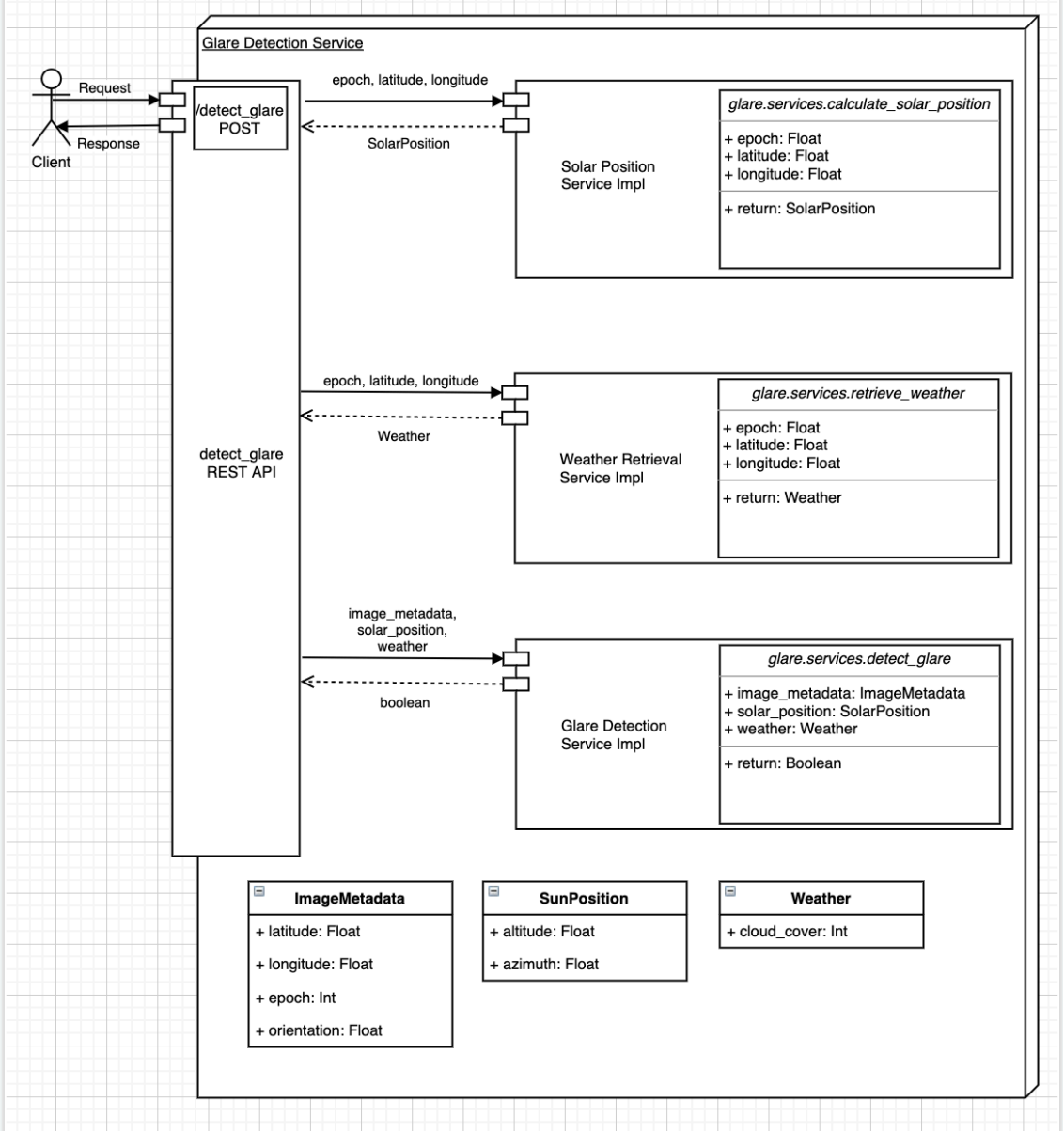


IMAGE METADATA COLLECTION

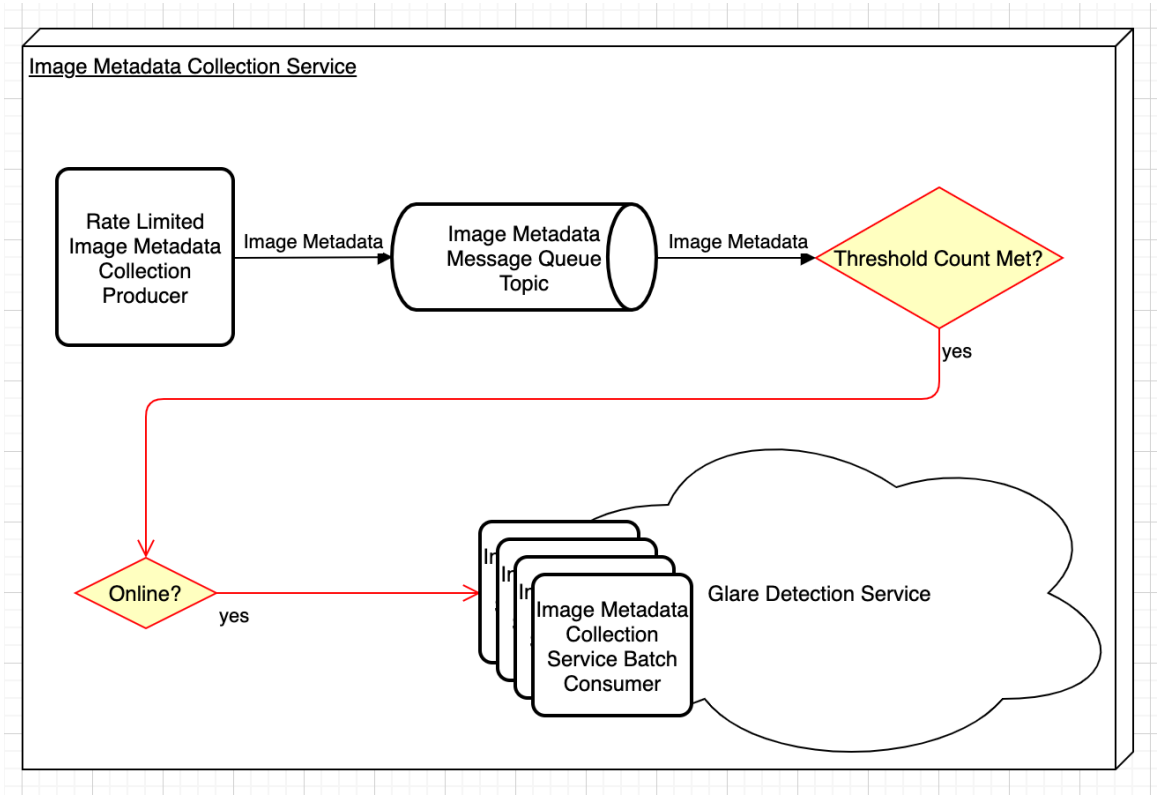
Service Requirements

We want to introduce a bulk API that can accept 1000's of data points collected from a test drive at frequency of 5 Hz and determine the glare condition for each data point.

Considerations

- While in transit, a car may randomly go offline (tunnels, remote areas, etc). We need a message queue to handle intermittent connectivity, rate limiting API calls and batching.
- We need to answer how tolerant we are of lost or dropped image metadata readings. If we tolerate lost messages, error handling on the message queue will likely be simple. If we can't, we'll need to introduce some offline cache to store messages while we wait to come back online.
- At a frequency of 5 data points collected per second, we will surpass 1000 data points collected every 3-4 minutes.
- Given that, as we only want to detect for glare every 6 minutes, each payload will only be a couple thousand data points.
- We only want to send the minimum amount of data possible per image (i.e. the metadata needed for detecting glare).

Service Architecture



- A long running process will exist on the vehicle to interface with the camera APIs, manage message queue production, and send that data to a message queue topic. The 5hz rate limiting is managed here.
- A message queue framework that supports sending multiple messages along in batches at a definable interval will be required to meet the budgetary API requests constraints.
- An image metadata collection service message queue consumer is introduced to the glare detection service to process incoming batches of image metadata.

SCALING AND DEPLOYMENT

Thousands of vehicles submitting thousands of messages simultaneously needs to be supported. As such, load balancing and

auto scaling is key here. To further support that, 2 horizontal scaling requirements are highlighted here:

- The image metadata message queue consumers
- The glare detection service

The glare detection service is containerized for deployment with the intention of using a container orchestrator such as Kubernetes that provides load balancing and autoscaling out of the box. The message queue consumer should be set up in Kubernetes to load balance by spinning up new consumers as needed based on load. From there, the glare detection service should be deployed independently in Kubernetes so it can also scale horizontally as needed. As these services are stateless, this should be possible as defined.

Gut instinct is to put them both in the same cluster but separate pods/services, but more investigation and load testing (e.g. with a tool like Gatling) is needed to determine the best deployment strategy.

<TODO: architectural diagrams showcasing this if needed>

OPEN QUESTIONS AND CONCERNS

- Security, auth, and identity management were not at all considered in this design. Will need to be before productizing.
- Once the glare detection service returns the results of the glare detection, what do we do with those?
    - Probably need persistence of some sort?
- On the API limit problem, we probably want to include some form of caching for 3rd party API weather data so we're not abusing that API access.
- What cloud cover level is still considered "sunny"?