Working solution

You can find solution deployed to Azure here:

https://weathermeasurements.azurewebsites.net/swagger/index.html

Examples:

https://weathermeasurements.azurewebsites.net/api/v1/devices/dockan/data/2016-01-13

https://weathermeasurements.azurewebsites.net/api/v1/devices/dockan/data/2016-01-13/humidity

API Requirements

I do not have detailed requirements for this API, so I tried to define them by myself using below questions.

What is the average use case of this API. Can we assume that historical data is accessed as frequently as data from last days?

Data could be requested for 3 reasons:

- calculate weather statistics from last few days, weeks or months
- calculate weather forecast based on data from last few days, weeks or months
- draw charts in web UI – usually for last month

Data older than 1 year is not accessed very often.

What does metadata file contain?

It contains list of all devices and sensors for which measurement data is available in blob storage. If data measurement for new device/sensor appears then new device and sensor is appended to metadata file.

How data is organized in historical file?

Measurements file are sorted by date, only newer files can be appended.

Let's assume there are missing files for some days in the middle of historical date range (there was some problem in delivery). Is it possible that this hole will be filled after some time?

No, only new files with later dates can be appended to historical file.

Is that possible that device can break, and then it will be fixed after some time and send data again?

Yes.

Should we expect many different devices and many different sensors per device?

There are no predictions, many devices can be connected to the system.

Will the data for each device be accessed locally or worldwide?

At the beginning service will provide data from devices placed nearby service. We are not expecting that data will be accessed from other continent.

Can we assume that csv measurement file for yesterday is complete no later than at 1 am CET and will be never modified in the future?

Yes.

How the data in measurement file is organized? Is there only one format in all files? Can this format change in the future?

Each measurement consist of timestamp and decimal value. Samples are sorted by timestamp. This is common format and we are not going to change it.

Design decisions:

Accessing historical file from blob storage each time when older data is requested is not optimal. I decided to store measurements in database that that will be accessed by REST API to read. In this case we will have better response time.

Elements:

SQL Database

SQL was used for simplicity, as I do not know what scale is needed. Measurements are stored per each [day, device, sensor]. There are day, device, sensor column for querying data and JSON to return data.

- Timestamps are stored without Date prefix to reduce amount of data.
- JSON is compressed with Brotli reduce time of transferring data and lower the cost of database usage. Fast method is used to reduce latency during decompression.

WeatherData.Api.Host Project

REST API for accessing imported data. In Memory Cache is used for single replica to lower latency for frequent requests.

WeatherData.Scheduler.Host Project

Scheduler will run import jobs every day to:

1. Check if new device or sensor appeared and import historical data
2. Import all files from last updated date (not only yesterday, in case there was some delivery to blob storage delay).

Possible future improvements

Database

If amount of stored data become very large, it will cause higher db costs and degraded query performance. We can think about few solutions to overcome this situation:

- Add scheduler job to remove data older than 1 year as they are used very rarely – in case this data is requested, historical file can be accessed and data can be imported for some short period of time
- Sharding, for example by device
- Use CosmosDB, as consistent hashing can easily help to scale and TTL could be used to remove old data

Scheduler

In case amount of device/sensor amount become large, scheduler can be scaled device/sensor, or even geographically if we have devices across multiple continents.

We can simply deploy multiple replicas of scheduler responsible for device/sensor or we can experiment with more asynchronous strategy:

1. Services for fetching data from blob storage
2. Imported data is pushed to the message queue, for example Azure Service Bus
3. Services subscribed to the queue are saving data to database,

Distributed Cache

In case average user needs to access data from multiple geo regions frequently and she/he needs lower latency, distributed cache (Redis) can be used.

In case we can define (based on usage statistics) some regular usage patterns – for example data from yesterday for devices located in Poland is used most frequently in Sweden - this data can be periodically prepopulated to distributed cache nodes nearby Sweden