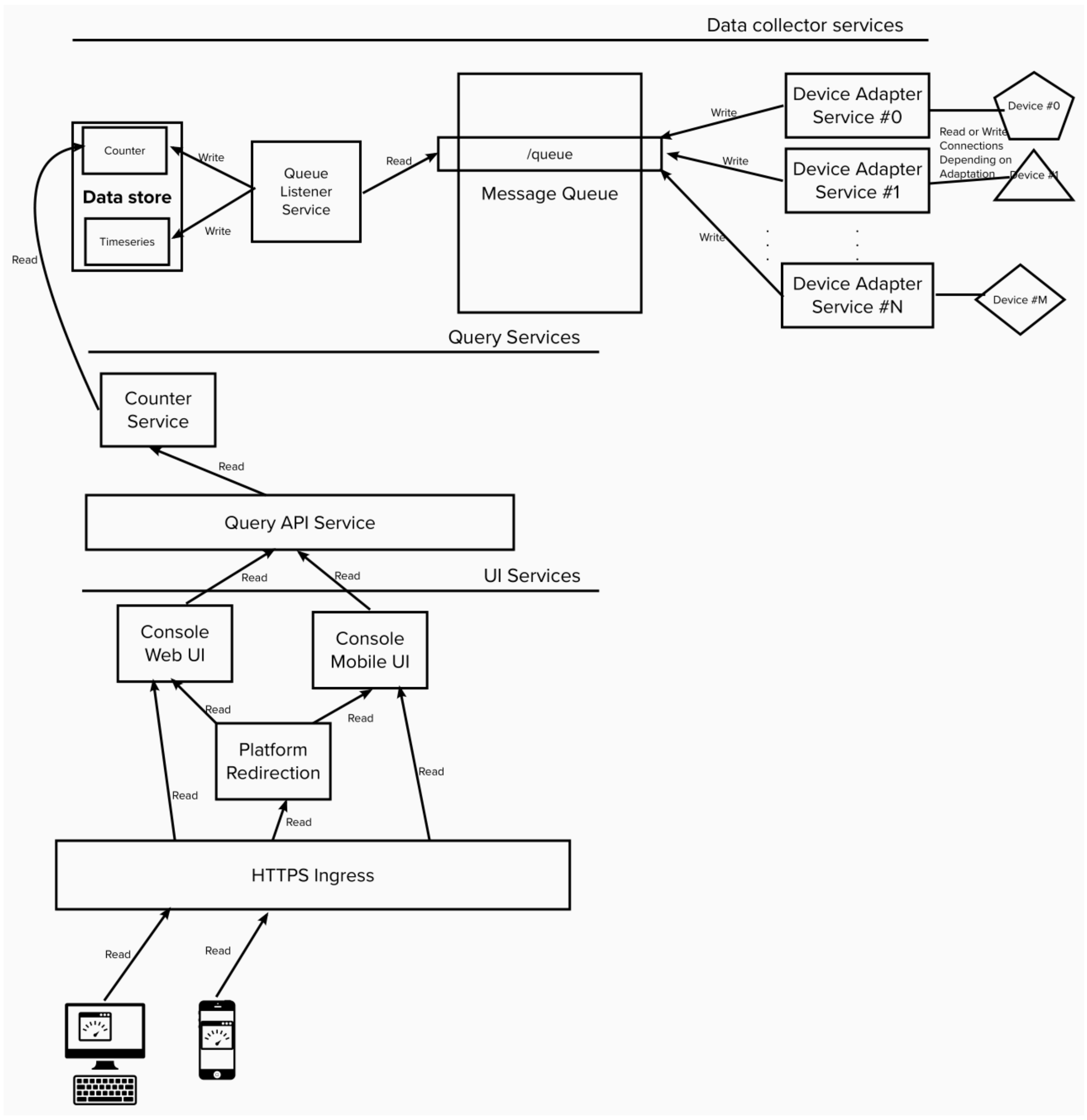
DDC Architecture

1. General Architecture

General Architecture could be divided into 3 service types: Data Collector, Query, and UI Services:



Data collector service components are: Device Adapter Services, Message Queue, Queue Listener Service, and Data Stores. These services take care of moving the data from devices up to final stores which will be persisted optimally for querying. Different storage formats & volumes could be considered for different query needs. And this multiple-persistence is fed by the Queue Listener. While Counter Store will be read by Counter Service, Timeseries Store is not read by any Query Services, however it's very important to store that important measurement data to be able to query it later for analytic/machine learning analysis.

Query Services serves query requests of the UI Services. Those services will reach at corresponding stores which were persisted by the Queue Listener and as those stores are optimized for particular queries it'd be very efficient to fulfill those requests. Query API Service is placed to provide consistent APIs that would guard UI from changing services at the back consequently easing service endpoint/format changes.

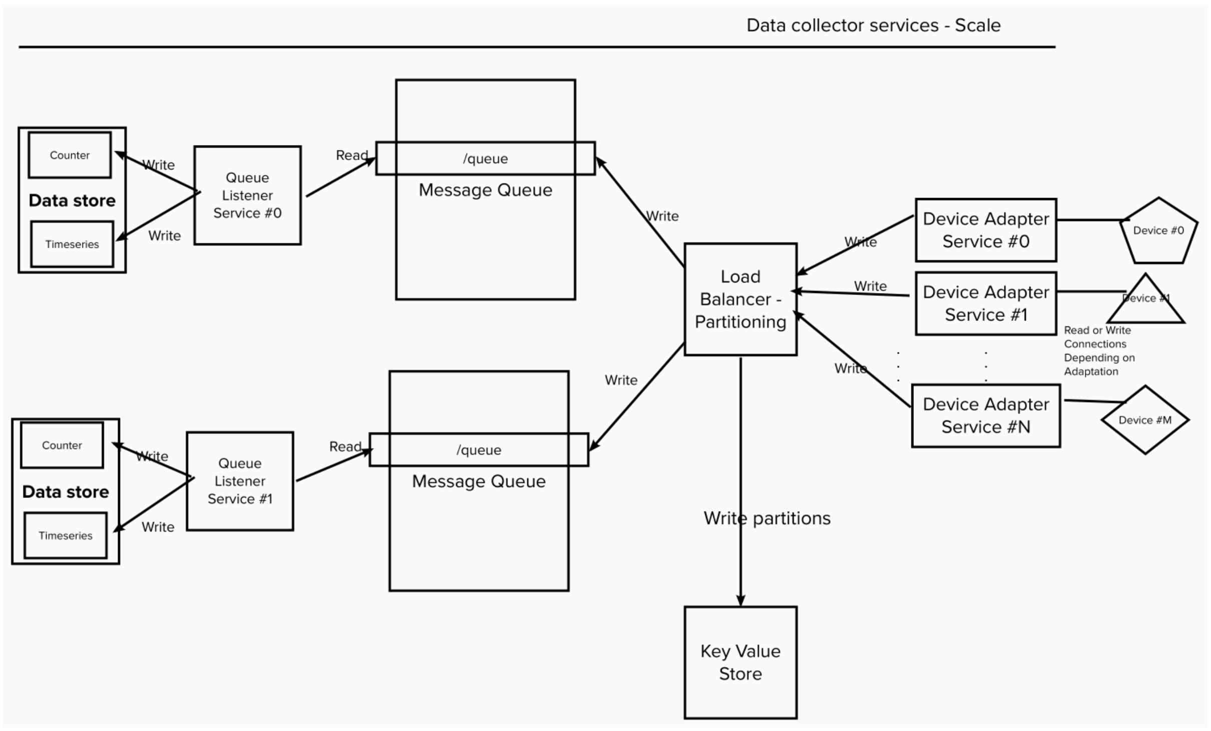
UI Services serve user requests: Console is the only UI application to provide. However it's vital to support mobile devices as well as desktop browsers. The services would be: Console Web UI, Console Mobile UI and Platform Redirector which redirects based on HTTP header.

2. Scaling Plan

Decision on scaling should be based on which component is loaded. If billions of devices are required to emit measurements thousands of times a minute, we must aim for Data Collector Services scaling. Whereas if the bottleneck is because of millions of users hitting web applications then UI services or Query Services scaling would be an option.

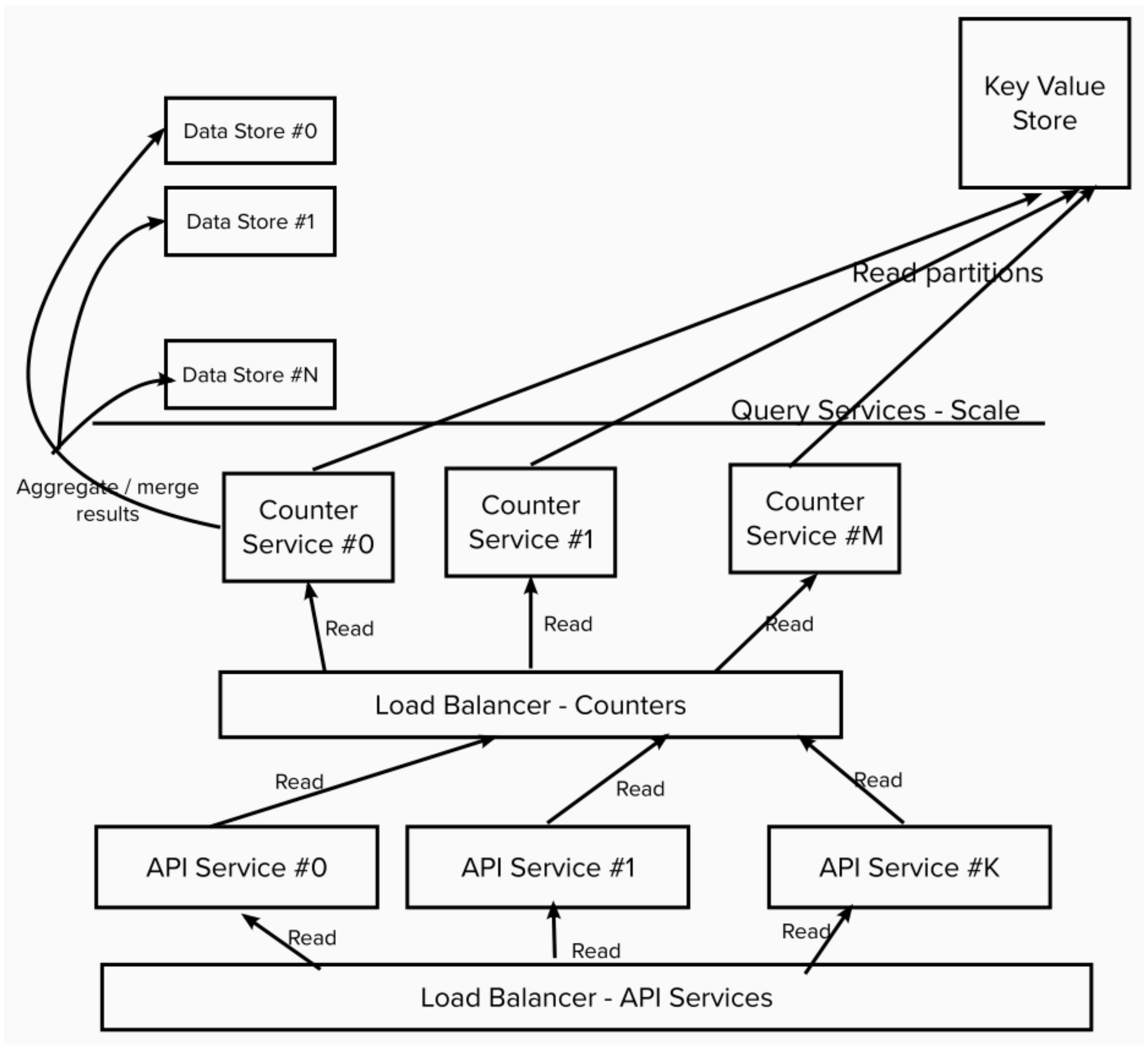
2.1 Data Collector Services Scaling

To be able to keep message ordering from the same device, it's suitable to support scaling by partitioning. Otherwise when one device's messages are fed to two different message queues, at the data store the original ordering could be lost due to network communications.


Data collector services - Scale
Counter
Data store
Timeseries
Write
Write
Queue Listener Service #0
Read
/queue
Message Queue
Write
Load Balancer - Partitioning
Write
Write
Write
Device Adapter Service #0
Device Adapter Service #1
Device Adapter Service #N
Device #0
Device #1
Device #M
Read or Write Connections Depending on Adaptation
Counter
Data store
Timeseries
Write
Write
Queue Listener Service #1
Read
/queue
Message Queue
Write
Write partitions
Key Value Store

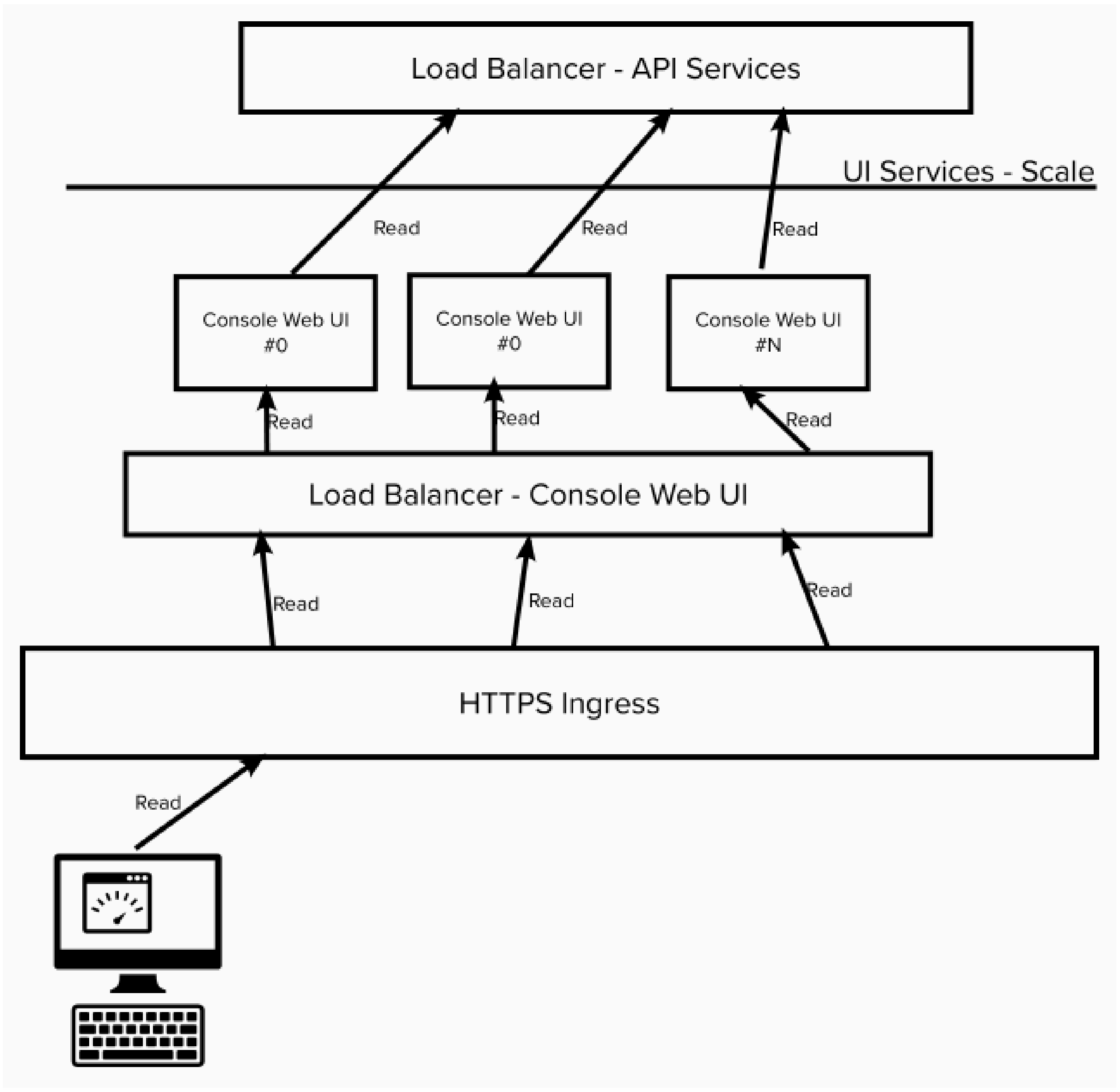
Note that Load Balancer is a special balancer which redirects measurement data based on Device measurements. A straightforward and balanced approach would be hash partitioning by Device ID. Note that the details of partitions are saved in the key value store.

2.2 Query Services Scaling
When Data Collector Services are partitioned, it's needed to track the number of partitions/stores to be able to merge them in Query Services. Since Query Services communicates with Data Stores only for reading, it's very easy to scale it by adding a load balancer (Load Balancer - Counters) in between.



As seen in the figure, API Service could scale just by spawning more API Service and allowing Load Balancer - API Services to distribute traffic evenly among them.

2.3 UI Services Scaling
UI Services scaling is trivial because they don't have side effects and access to the Query Services read-only. It's a matter of spawning new services and having load balancer to spread the load.



Here, only Console Web UI scaling is depicted but it’s similar for other services.