Context:

This is related to the issue: https://github.com/apache/incubator-pinot/issues/4484

We've noticed some cases where some of the Pinot servers are undergoing severe GC pauses. This leads to a subset of queries to timeout or throw an exception (did not hear from all servers). This is because the Pinot broker randomly selects one of the replicas responsible for a given segment to route the query. Currently, it has no knowledge of which replicas are "unhealthy" and should be avoided. This proposal describes adding a health checker to the Pinot Broker to detect such unhealthy servers and temporarily take them out of rotation.

What is an unhealthy server ?

An unhealthy server is one which is still active - i.e. it exists in the Helix external view but is unable to serve any request. This might happen if the server is going through full GC repeatedly.

High level architecture

Health Checker
A new Health Checker component will be added within a Pinot Broker. It tracks the "unhealthy" subset of servers that should not be used for routing requests. The minimal interface looks something like this:

```
public interface ServerHealthChecker {
        boolean isUnhealthy(String instanceName);
        Set<String> getUnhealthyServers();
        // This method may be a NOOP for some implementations
        void registerResponse(ServerResponse response);
}
```

Implementations will need to obtain the list of server instance names (eg: from Helix External View) and track node health for each such instance. The exact mechanism for tracking node health can differ based on the detection algorithm.

Here are 2 considerations:

1) Response based:

For each instance, look at the previous responses obtained from that particular instance. This can be obtained from ‘ServerResponse’ received during the query scatter gather. If the query response time has been consistently higher or a lot of timeouts have been observed, then this instance can be marked unhealthy. There are different ways of computing consistently unhealthy query response:

- One of the most accurate ways could be to use a histogram over a rolling window of some size (eg: 5 minutes). Then we can compare the p50 value with our threshold (eg: something like 10 seconds).
- We could also only look at the last N contiguous responses and check if they’re all falling outside the healthy range.

In addition to this, we’ll need to keep track of exceptions (for eg: when the server times out). All these values together can be used to mark the node unhealthy.

Resetting the Node:
After a certain time period, this node status should be reset to healthy so that queries can be resumed to this node and we can repeat the cycle all over again. This is using an optimistic approach giving the corresponding set of unhealthy servers to recover on their own.

2) Heartbeat based

Periodically poll each server endpoint to check if the server is responding to requests. The number of unhealthy responses (i.e. high turnaround time) and / or error rate can be used to determine node health. Similarly, polling will also determine if a node previously marked unhealthy has recovered and health status can be reset.

Using the Health Checker

The Broker can use this to prune such unhealthy servers inside the ‘select’ method (InstanceSelector). In case, all servers replicas responsible for a given segment are marked as unhealthy - then we can return an error.