Report

Design details

- The implementation is divided into 3 major components.

    1. Agent
    2. Reporter
    3. Source

    `Source` interface represent a metrics source. It has two implementation, `http` and `system` source. `http` represents the endpoints used to fetch the metrics from algod-node. `system` source represents the algod-node process-stat related metrics.

    `Reporter` interface exposes methods to report all the collected metrics. Its implementation `fileReporter` keeps on receiving metrics from different source into a channel and internally stores them in a buffer. A buffer is used here to avoid overhead of write-syscalls during writing data to a file. All the collected metrics are flushed to output file after the buffer is full or the agent shuts down.

    `Agent` orchestrates all the above components. It starts a `Reporter` in a goroutine to collect metrics. To maximise the throughput, each `Source` performs the metrics collection task in a goroutine. Based on the sampling-frequency interval, the `Agent` call all the `Source` for metrics collection. In case of abrupt shutdown of `Agent`, a graceful shutdown happens. It includes flushing the buffer to the output file, closing opened files, channels and wait for goroutines to finish.

- Parsing the `http` source depends on the `Content-Type` returned. Here the assumption is made that if the content is of type `text/plain`, then it's already in metrics data-model format and no parsing is required. For the returned json content, only integer/float value are considered as metric candidates(Gauge).
- `system` source metrics are considered to be of type `Gauge` metric. This source also handles the case where the processID(pid) of algod-node changes. It re-sync the algo-node PID, if it changes during runtime.
- For this implementation I have considered sampling-frequency of agent to be at least 1 second.
- A configuration should be supplied while running the `Agent`. See below example `yaml` configuration:

```
 # specify all the http metrics source
httpsources:
 - endpoints: http://localhost:8080/metrics
   headers:
    x-api-key : abcd
 - endpoints: http://localhost:8080/metricsv2/status
   headers:
    x-api-key : abcd

# sample frequency is in seconds
sampleFrequency: 5

# output file name
targetOutputFile: output_file
```

Even if no `httpsources` are specified in the configuration, `Agent` will still collect process-stat metrics. Specify the required http-headers for http-source like API-keys or Authentication-header.

Enhancement scope

Current implementation is suitable for POC purpose, below are some of the enhancements which can be done to make it production ready:

- New metric-source can be easily added by implementing `Source` interface.
- If we want to persist collected metrics in a database or forward it to some other component, it can be achieved by extending the `Reporter` interface.
- Current implementation rely on a buffer of size 4KB to collect metrics. But with more metrics, a bigger size buffer must be used.
- To reduce the size of output file suitable compression techniques can be used. Also, down-sampling the data can be done to avoid data redundancy.
- If http-sources are returning error continuously, we can introduce backoff-retry mechanism in the Agent to handle it.
- Cpu/memory profiling can be done to find bottlenecks/issues and remove them.