Feature Project Proposal——Akraino Profiling

Helloway He
(helloway.wewe@gmail.com)
04/03/2019

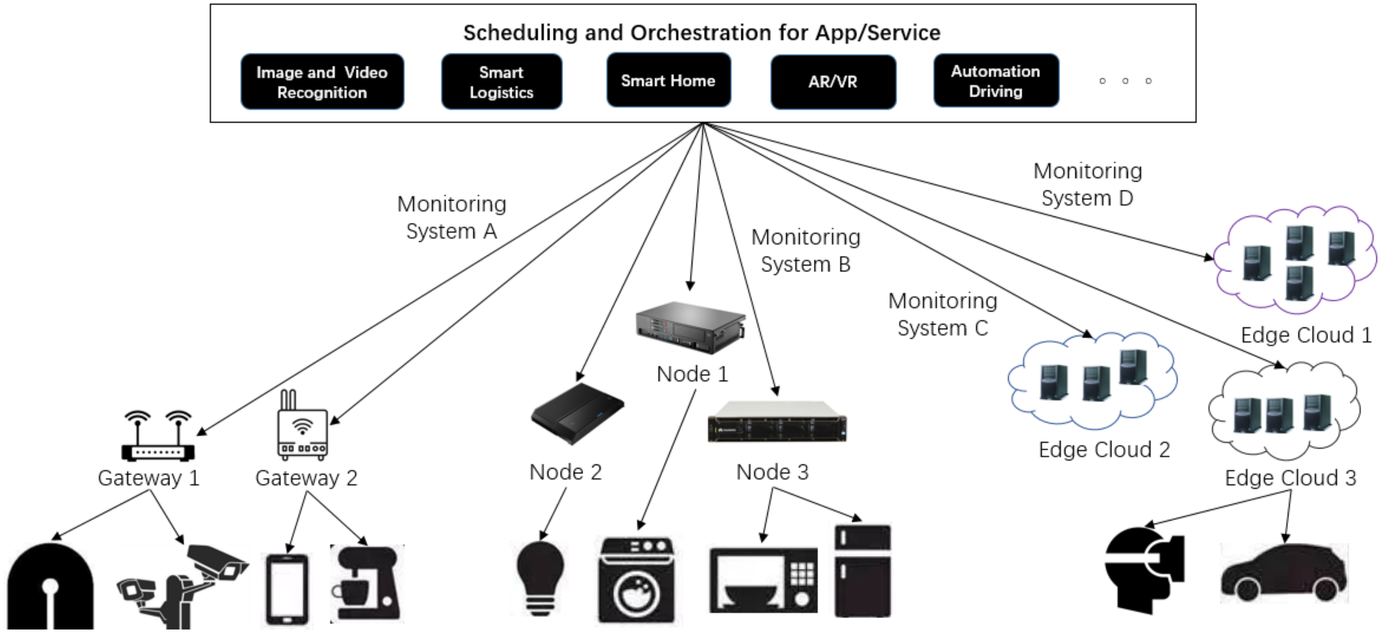
AKRAINO
EDGE STACK

Background - Requirement for Unified Metrics Exposition Format

- Variety of edge computing devices – How to choose the most suitable device to run the edge computing apps/services
- Various monitoring and management systems, different exposed metrics data formats of backend – Hard to pick out the optimal one from the competing products for suppliers
  - Give priority to supporting official de facto standard
  - How to seamlessly connect all other monitoring systems to gather more metrics
  - How to reduce development complexity

These questions require us to

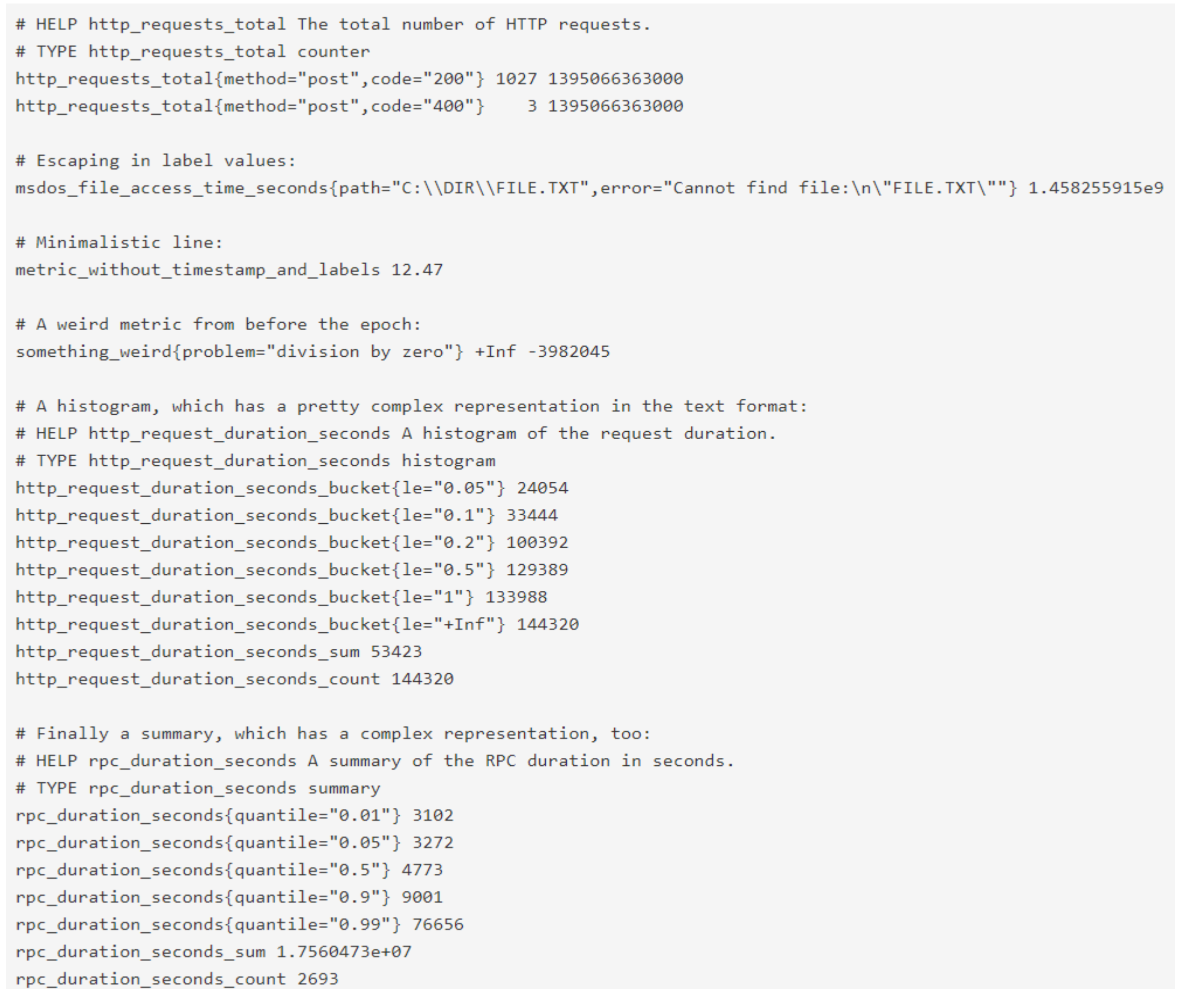
- Have the widely accepted standard that can expose the metric data collected by the backend in a unified format
- Integrate and support the common data format to enable seamless multi-system docking - More comprehensive metrics data, better scheduling



Diversified Equipment and Monitoring Systems
No Unified Standard for Exposing Metric Data Collected
Hard to Schedule and Orchestrate Edge App/Service

AKRAINO EDGE STACK

Background – OpenMetrics Metrics Exposition Format Standard



```
# HELP http_requests_total The total number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{method="post",code="200"} 1027 1395066363000
http_requests_total{method="post",code="400"}    3 1395066363000

# Escaping in label values:
msdos_file_access_time_seconds{path="C:\\DIR\\FILE.TXT",error="Cannot find file:\n\"FILE.TXT\""} 1.458255915e9

# Minimalistic line:
metric_without_timestamp_and_labels 12.47

# A weird metric from before the epoch:
something_weird{problem="division by zero"} +Inf -3982045

# A histogram, which has a pretty complex representation in the text format:
# HELP http_request_duration_seconds A histogram of the request duration.
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{le="0.05"} 24054
http_request_duration_seconds_bucket{le="0.1"} 33444
http_request_duration_seconds_bucket{le="0.2"} 100392
http_request_duration_seconds_bucket{le="0.5"} 129389
http_request_duration_seconds_bucket{le="1"} 133988
http_request_duration_seconds_bucket{le="+Inf"} 144320
http_request_duration_seconds_sum 53423
http_request_duration_seconds_count 144320

# Finally a summary, which has a complex representation, too:
# HELP rpc_duration_seconds A summary of the RPC duration in seconds.
# TYPE rpc_duration_seconds summary
rpc_duration_seconds{quantile="0.01"} 3102
rpc_duration_seconds{quantile="0.05"} 3272
rpc_duration_seconds{quantile="0.5"} 4773
rpc_duration_seconds{quantile="0.9"} 9001
rpc_duration_seconds{quantile="0.99"} 76656
rpc_duration_seconds_sum 1.7560473e+07
rpc_duration_seconds_count 2693
```

OpenMetrics Metrics Data Format

- CNCF sandbox project launched by the Prometheus community, Prometheus is the second project graduated from CNCF after Kubernetes
- Main contributors
  - Google, Prometheus, InfluxData, SolarWinds, Open Census, Uber, Data Dog, etc.
- Aims to create a standard specifically for exposing metric data
- Uses the Prometheus exposition format as the starting point for its standard, evolve Prometheus' metrics format and semantics into a recognized de facto standard specification
- Data model supports multi-dimensional definition
  - Metrics format - Time series data consist of the name of the metric and a series of labels (key-value pairs)

    <metric name>{<label name>=<label value>, ...}, For example: Http_requests_total{method="POST", code=“200"}.
  - Metrics type
    - **Counter** a cumulative metric that only can increase, things like the number of requests served
    - **Gauge** a metric that can go up and down, ,things like temperature or current memory usage.
    - **Histogram** usually things like request durations or response sizes
    - **Summary** like Histogram, calculate quantiles

AKRAINO EDGE STACK

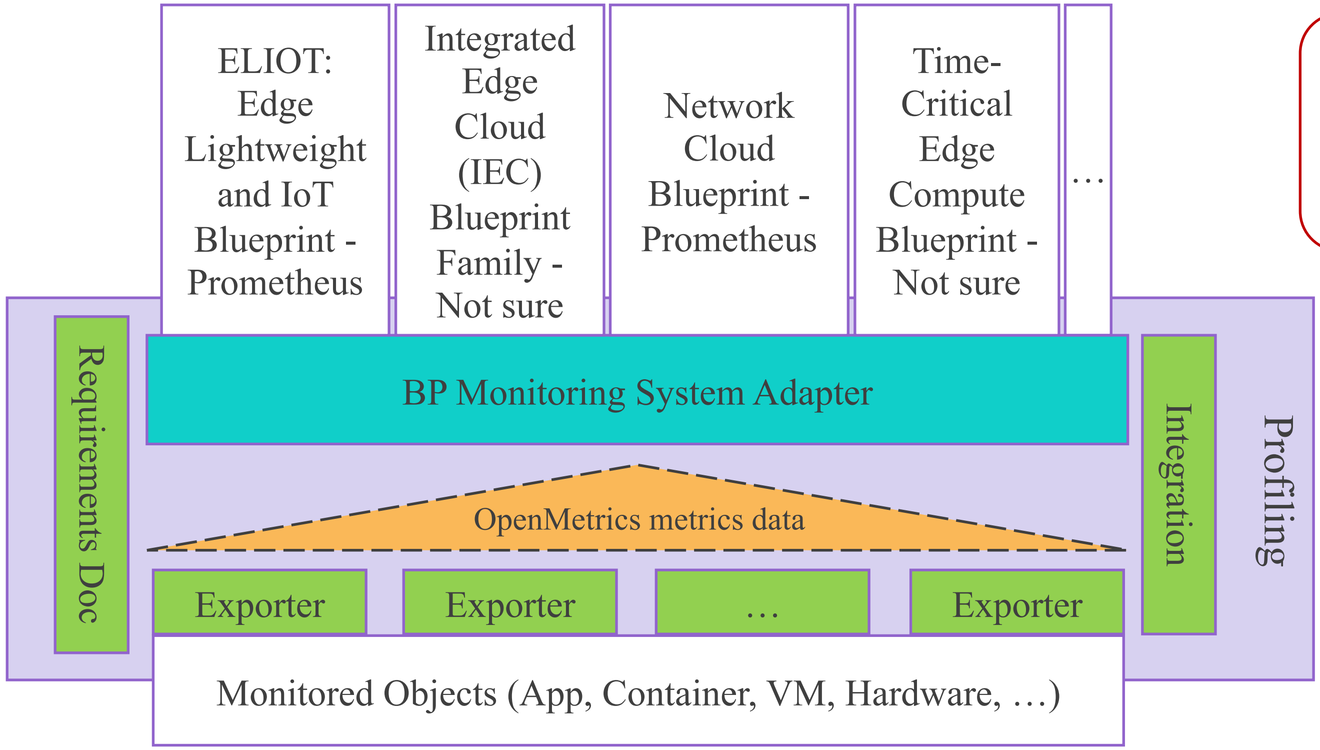
Feature Project Proposal - Akraino Profiling

Feature	Description	Companies Participating / Committers	Requested Release / Timeline	Informational
Akraino Profiling	Akraino Profiling to provide the exposed metrics data that are compliant with the OpenMetrics standard to unify the end-to-end metric format for edge computing scenarios. 1. Pre-analysis Analyze which metrics data required to be monitored, then aggregate them into the requirement document; (Preliminarily plan to focus on IOT scenario). 2. Exporter -- Collect metrics data from the monitored object and expose them to the monitoring systems in OpenMetrics standard format. (Compliant with the OpenMetrics) Analyze which exporters need to be add/redevelop/updated according to the metrics requirement document, then develop them 3. Adapter (optional, pluggable and dynamically loaded component) -- Convert OpenMetrics metrics data exposed by exporters to their own proprietary format that can be directly processed by those monitoring systems. (Compatible with OpenMetrics) Analyze which BP monitoring systems need to develop the adapter: • If the BP monitoring system can't directly process the metric data which follow the OpenMetrics format standard, it needs to develop the adapter to do metric format transformation to dock with it. • Otherwise, there is no need to develop the adapter for the monitoring system of the blueprint that can directly process the OpenMetrics metric data. 4. Integration To develop scripts based on ansible/helm to integrate Profiling into each BP, i.e., deploy the exporters and converters that BP needs, and then validate in the Akraino environment. 5. Future plans to extend to more Akraino Blueprint Families/BPs.	Huawei ARM Intel	R2	Impacted Blueprint Family – Applies to all BP Families and Blueprints See next slide for additional details

AKRAINO EDGE STACK

Akraino Profiling Framework

- **Add/redevelop/update exporters for them to collect metrics data from the monitored objects and expose in OpenMetrics format.**
- **Adapter is the optional component**
  1. For those monitoring systems that are not designed for processing the OpenMetrics metric data -- Develop the corresponding adapter for them to transform OpenMetrics metric data format exposed by exporters to their own proprietary format that can be directly processed by themselves.
  2. For those monitoring systems that are designed for processing the OpenMetrics metric data **--** No need to develop adapter for them as they can directly process the OpenMetrics data from exporters.

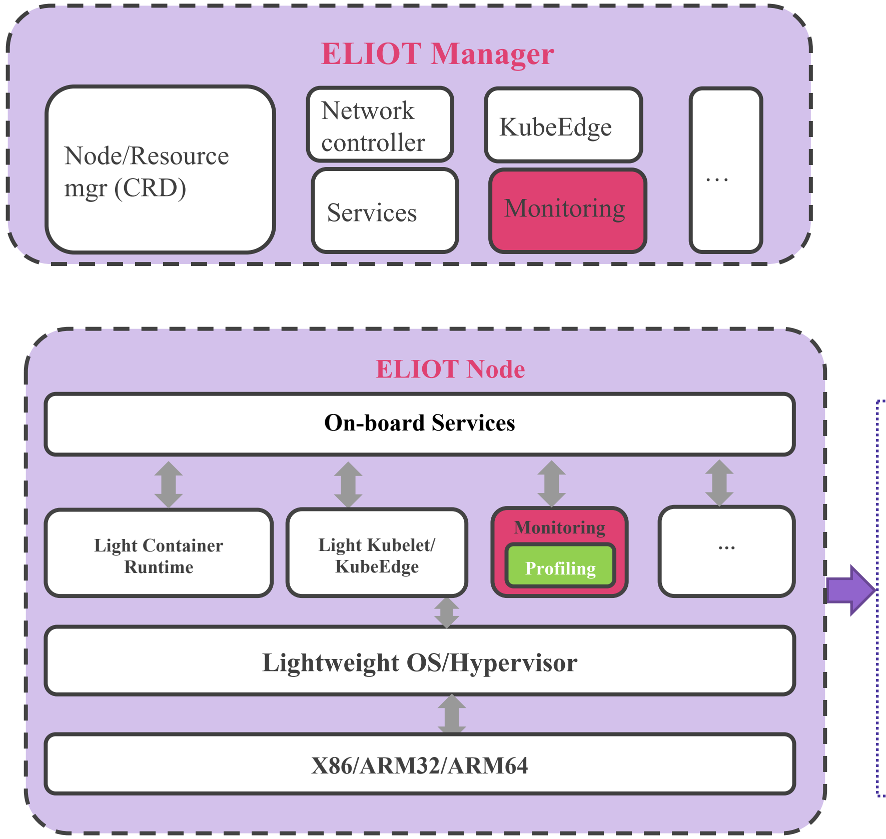

ELIOT: Edge Lightweight and IoT Blueprint - Prometheus
Integrated Edge Cloud (IEC) Blueprint Family - Not sure
Network Cloud Blueprint - Prometheus
Time-Critical Edge Compute Blueprint - Not sure
…
Requirements Doc
BP Monitoring System Adapter
OpenMetrics metrics data
Exporter
Exporter
…
Exporter
Integration
Profiling
Monitored Objects (App, Container, VM, Hardware, …)

Akraino Profiling consists of:
- Requirement Doc – Required;
- Exporter – Required;
- Adapter – Optional;
- Integration – Required;

AKRAINO EDGE STACK

Akraino Profiling Example 1 - ELIOT

- Pre-analysis
  Monitoring Metrics Data: eliot_node1_sound, eliot_node1 _temperature,… The requirements document is roughly as follows.
- Exporter
  Add a new node exporter/update an existing node exporter to collect and expose metrics data which are compliant with OpenMetrics standard;
- Adapter (Eliot Not required)
  Eliot uses Prometheus as the monitoring system that can directly process the OpenMetrics metric data collected;
- Integration
  Develop scripts based on ansible/helm to integrate Profiling with Eliot.



```
# HELP eliot_node1_sound Sound (noise) level.
# TYPE eliot_node1_sound gauge
eliot_node1_sound 1
# HELP eliot_node1 _temperature Temperature in C and F.
# TYPE eliot_node1 _temperature gauge
eliot_node1_temperature{metric="celsius"} 19.093447311567388
eliot_node1_temperature{metric="fahrenheit"} 66.3682051608213
…
```

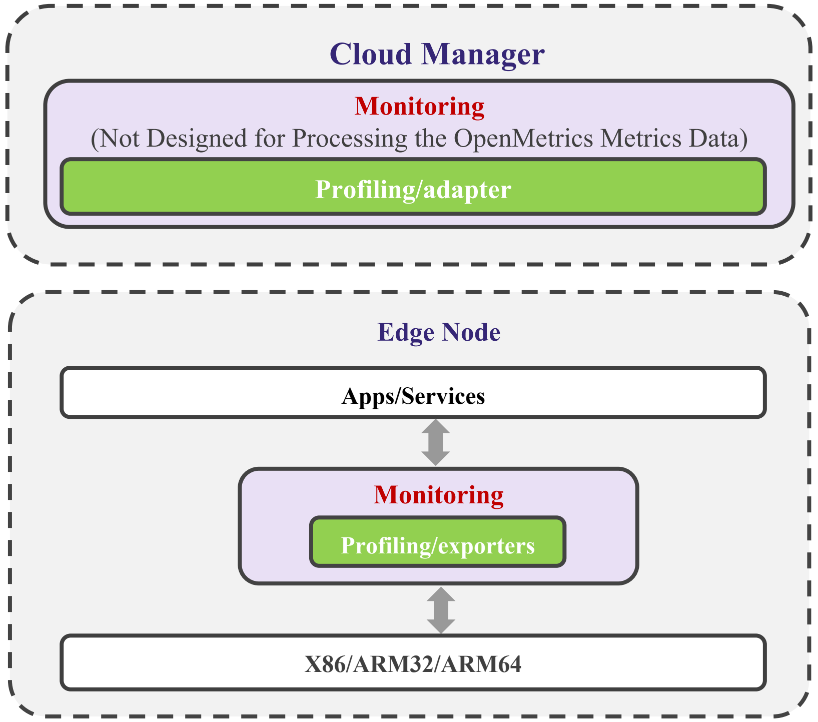
Monitoring Metric Data

Metric Name	Metric Type	Metric Description
eliot_node1_sound	gauge	Sound (noise) level
eliot_node1 _temperature	gauge	Temperature in C and F

Monitoring (developer/user tools)
Using prometheus and cadvisor to monitoring resources
Profiling
Add/update **exporters** to collect and expose the metric data to Prometheus server on the Eliot Manager in the OpenMetrics format

AKRAINO EDGE STACK

Akraino Profiling Example 2



Monitoring (developer/user tools)
Using Ganglia, Nagios, etc., and their own collector daemon to monitoring resources

Profiling

1. **Edge Node** Add/redevelop **exporters** to collect and expose the metric data to the monitoring system on the Cloud Manager in the OpenMetrics format.
2. **Cloud Manager** Develop **adapter** to convert OpenMetrics metric data format to proprietary format that they can process.

Take Ganglia (on the cloud) and gmond (Ganglia collector daemon at the edge) as an example:

- Pre-analysis
  Determined the monitoring metrics;
- Exporter
  Redevelop gmond_exporter to serve OpenMetrics metrics for ganglia on the Cloud;
- Adapter (Ganglia required)
  Develop ganglia_adapter for ganglia to convert the OpenMetrics metric to ganglia metrics as it can't directly process the OpenMetrics metric collected by gmond_exporter;
- Integration
  Develop scripts based on ansible/helm to integrate Profiling which includes cloud part and edge part.

AKRAINO EDGE STACK

4/23/19