Streaming Telemetry Under the Hood: Something to Think About

Viktor Osipchuk (vosipchu@cisco.com)
05-10-2018
Do They Select Cars By Color?
Agenda

1. Brief Telemetry Overview
2. Closer Look at Router
3. Closer Look at the Link Between
4. Closer Look at Collector
5. Final Thoughts
Telemetry: an automated communications process by which measurements and other data are collected at remote or inaccessible points and transmitted to receiving equipment for monitoring.
https://en.wikipedia.org/wiki/Telemetry
Agenda

1. Brief Telemetry Overview
2. Closer Look at Router
3. Closer Look at the Link Between
4. Closer Look at Collector
5. Final Thoughts
How Do You See Telemetry In a Router?

Exporter layer
- Encoding and transportation for the models

Producer layer
- Time intervals definitions for the models

Data model layer
- Raw data mapped to a model (YANG native, OpenConfig, etc.)

Data store layer
- Native (raw) data inside a router's database
Check Your Models

Native Model

Open Model

Native device config/oper data

Mapped config/oper data

Deviations

This might not always work for Telemetry
What Does Sample Interval Really Mean?

- Works fine for small collections
- You should never see missed collections
- Hard to automate, no consistent behavior

This might vary in time

- Consistent behavior, easy to automate
- Sample interval must be more than max collection time
- You might see missed collections

Works fine for small collections
You should never see missed collections
Hard to automate, no consistent behavior

Consistent behavior, easy to automate
Sample interval must be more than max collection time
You might see missed collections

This might vary in time
Missed Collections. What To Do?

What do you expect from your router?

- Send all 1s / all 0s?
- Send previous collection values?
- Leave things as is and don’t send anything?

What will your router do here?

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>CPU</th>
<th>(Epoch converted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>152293666977000000</td>
<td>9</td>
<td>(April 5, 2018 1:57:49.977 PM)</td>
</tr>
<tr>
<td>152293667497700000</td>
<td>9</td>
<td>(April 5, 2018 1:57:54.977 PM)</td>
</tr>
<tr>
<td>152293667997900000</td>
<td>9</td>
<td>(April 5, 2018 1:57:59.979 PM)</td>
</tr>
<tr>
<td>152293668497600000</td>
<td>9</td>
<td>(April 5, 2018 1:58:04.976 PM)</td>
</tr>
<tr>
<td>152293668997800000</td>
<td>9</td>
<td>(April 5, 2018 1:58:09.978 PM)</td>
</tr>
<tr>
<td>152293669497300000</td>
<td>9</td>
<td>(April 5, 2018 1:58:14.973 PM)</td>
</tr>
<tr>
<td>152293669983000000</td>
<td>9</td>
<td>(April 5, 2018 1:58:19.983 PM)</td>
</tr>
<tr>
<td>152293670497600000</td>
<td>9</td>
<td>(April 5, 2018 1:58:24.976 PM)</td>
</tr>
<tr>
<td>152293670997600000</td>
<td>9</td>
<td>(April 5, 2018 1:58:29.976 PM)</td>
</tr>
<tr>
<td>152293671497600000</td>
<td>9</td>
<td>(April 5, 2018 1:58:34.976 PM)</td>
</tr>
<tr>
<td>152293671997500000</td>
<td>9</td>
<td>(April 5, 2018 1:58:39.975 PM)</td>
</tr>
<tr>
<td>152293672497500000</td>
<td>9</td>
<td>(April 5, 2018 1:58:44.975 PM)</td>
</tr>
<tr>
<td>152293672998200000</td>
<td>9</td>
<td>(April 5, 2018 1:58:49.982 PM)</td>
</tr>
<tr>
<td>152293673497600000</td>
<td>8</td>
<td>(April 5, 2018 1:58:54.976 PM)</td>
</tr>
</tbody>
</table>

Do you want to fill the gap or not?
How Do You Want Your Data Out?

Transport
- gRPC
- TCP
- UDP

Encoding
- GPB
- KV GPB
- JSON

Models
- Open / Openconfig
- Proprietary / Native
Is It Enough To State gRPC/GPB Support?

Define your Telemetry message

```protobuf
syntax = "proto3";
option go_package = "telemetry_bis";

/* Common Telemetry message */  // this is common for both
message Telemetry {
  oneof node_id {
    string node_id_str = 1;
  }
  oneof subscription {
    string subscription_id_str = 3;
  }
  string encoding_path = 6;
  uint64 collection_id = 8;
  uint64 collection_start_time = 9;
  uint64 msg_timestamp = 10;
  repeated TelemetryField data_gpbkv = 11;
  TelemetryGPBTable data_gpb = 12;
  uint64 collection_end_time = 13;
}
```


Define your gRPC calls

```protobuf
define your gRPC calls

service gRPCConfigOper {
  // Configuration related commands
  rpc GetConfig(ConfigGetArgs) returns(stream ConfigGetReply) {};
  rpc MergeConfig(ConfigArgs) returns(ConfigReply) {};
  rpc DeleteConfig(ConfigArgs) returns(ConfigReply) {};
  rpc ReplaceConfig(ConfigArgs) returns(ConfigReply) {};
  rpc CliConfig(CliConfigArgs) returns(CliConfigReply) {};
  rpc CommitReplace(CommitReplaceArgs) returns (CommitReplaceReply) {};
  rpc CommitConfig(CommitArgs) returns(CommitReply) {};
  rpc ConfigDiscardChanges(DiscardChangesArgs) returns(DiscardChangesReply) {};
  // Get only returns oper data
  rpc GetOper(GetOperArgs) returns(stream GetOperReply) {};

  // Get Telemetry Data
  rpc CreateSubs(CreateSubsArgs) returns(stream CreateSubsReply) {};
}
```

Is It Enough To State gRPC/GPB Support?

Juniper’s OC .proto

```proto
package telemetry;

// Interface exported by Agent
service OpenConfigTelemetry {
  // Request an inline subscription for data at the specified path
  // The device should send telemetry data back on the same
  // connection as the subscription request.
  rpc telemetrySubscribe(SubscriptionRequest)
    returns (TelemetryFieldOptions);

  // Terminates and removes an existing telemetry subscription
  rpc cancelTelemetrySubscription(CancelSubscriptionRequest)
    returns (Empty);

  // Get the list of current telemetry subscriptions from the
  // target. This command returns a list of existing subscriptions
  // not including those that are established via configuration.
  rpc getTelemetrySubscriptions(GetSubscriptionsRequest)
    returns (TelemetryFieldOptionsList);

  // Get Telemetry Agent Operational States
  rpc getTelemetryOperationalState(GetOperationalStateRequest)
    returns (TelemetryFieldOptions);

  // Return the set of data encodings supported by the device for
  // telemetry data
  rpc getDataEncodings(DataEncodingRequest)
    returns (DataEncodingResponse);
}
```


Juniper’s UDP .proto

```proto
import "pbj.proto";
import "google/protobuf/descriptor.proto";

extend google.protobuf.FieldOptions {
  optional TelemetryFieldOptions telemetry_options = 1024;
}

message TelemetryFieldOptions {
  optional bool is_key = 1;
  optional bool is_timestamp = 2;
  optional bool is_counter = 3;
  optional bool is_gauge = 4;
}

message TelemetryStream {
  // router name or export IP address
  required string system_id = 1 "((telemetry_options).is_key = true,
      (pbj_field_option).type = FT_POINTER);"

  // line card / RE (slot number)
  optional uint32 component_id = 2 "((telemetry_options).is_key = true);"

  // PFE (if applicable)
  optional uint32 sub_component_id = 3 "((telemetry_options).is_key = true);"

  // configured sensor name
  optional string sensor_name = 4 "((telemetry_options).is_key = true,
      (pbj_field_option).type = FT_POINTER);"
}
```

[https://github.com/nileshsimaria/jmonudp/blob/master/protos/telemetry_top/telemetry_top.proto](https://github.com/nileshsimaria/jmonudp/blob/master/protos/telemetry_top/telemetry_top.proto)
GNMI Should Be The Answer. Right?

Defines the gRPC call

```plaintext
service GNMI {
    // Capabilities allows the client to retrieve the set of capabilities that
    // is supported by the target. This allows the target to validate the
    // service version that is implemented and retrieve the set of models that
    // the target supports. The models can then be specified in subsequent RPCs
    // to restrict the set of data that is utilized.
    // Reference: GNMI Specification Section 3.2
    rpc Capabilities(CapabilityRequest) returns (CapabilityResponse);
    // Retrieve a snapshot of data from the target. A Get RPC requests that the
    // target snapshots a subset of the data tree as specified by the paths
    // included in the message and serializes this to be returned to the
    // client using the specified encoding.
    // Reference: GNMI Specification Section 3.3
    rpc Get(GetRequest) returns (GetResponse);
    // Set allows the client to modify the state of data on the target. The
    // paths to modified along with the new values that the client wishes
    // to set the value to.
    // Reference: GNMI Specification Section 3.4
    rpc Set(SetRequest) returns (SetResponse);
    // Subscribe allows a client to request the target to send it values
    // of particular paths within the data tree. These values may be streamed
    // at a particular cadence (STREAM), sent one off on a long-lived channel
    // (POLL), or sent as a one-off retrieval (ONCE).
    // Reference: GNMI Specification Section 3.5
    rpc Subscribe(stream SubscribeRequest) returns (stream SubscribeResponse);
}
```

Defines the message

```plaintext
message Update {
    Path path = 1;  // The path (key) for the update.
    Value value = 2 [deprecated=true];  // The value (value) for the update.
    TypedListValue val = 3;  // The explicitly typed update value.
    uint32 duplicates = 4;  // Number of coalesced duplicates.
}
```

What to select here?

https://github.com/openconfig/gnmi/blob/master/proto/gnmi/gnmi.proto

Agenda

1. Brief Telemetry Overview
2. Closer Look at Router
3. Closer Look at the Link Between
4. Closer Look at Collector
5. Final Thoughts
How Can a Router Send Its Data Out?

- gRPC
- TCP
- UDP
gRPC Comes With an Overhead...

- Magic number to start HTTP2 phase
- Settings from the router
- Window size from the router
- HTTP2 details
- Window size/settings from the collector
- Settings confirmations

Negotiation happens before the data is streamed
...But Brings Some Good Benefits

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16.020449012</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>79</td>
<td>WINDOW_UPDATE</td>
</tr>
<tr>
<td>4</td>
<td>16.020453129</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>16459</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.02046328</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>79</td>
<td>WINDOW_UPDATE</td>
</tr>
<tr>
<td>4</td>
<td>16.020469649</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>16459</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.020476567</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>1514</td>
<td>64969 → 57500</td>
</tr>
<tr>
<td>4</td>
<td>16.020620532</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>66</td>
<td>57500 → 64969</td>
</tr>
<tr>
<td>4</td>
<td>16.021387959</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>66</td>
<td>64945 → 57500</td>
</tr>
<tr>
<td>4</td>
<td>16.021410692</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>16459</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021419367</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>10293</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021421080</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>527</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021424808</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>237</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021430133</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>679</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021472771</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>79</td>
<td>WINDOW_UPDATE</td>
</tr>
<tr>
<td>4</td>
<td>16.021480055</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>1048</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021509967</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>676</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021513831</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>66</td>
<td>57500 → 64945</td>
</tr>
<tr>
<td>4</td>
<td>16.021583417</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>670</td>
<td>DATA</td>
</tr>
<tr>
<td>4</td>
<td>16.021684650</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>66</td>
<td>57500 → 64945</td>
</tr>
<tr>
<td>4</td>
<td>16.021787475</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>HTTP2</td>
<td>79</td>
<td>WINDOW_UPDATE</td>
</tr>
<tr>
<td>4</td>
<td>16.021836970</td>
<td>10.30.110.40</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>1514</td>
<td>64969 → 57500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flags: 0x00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0...</td>
</tr>
<tr>
<td>.000 0000</td>
</tr>
<tr>
<td>.000 0000</td>
</tr>
<tr>
<td>.000 0000</td>
</tr>
</tbody>
</table>
TCP And UDP Are Simple

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Leng</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.000000000</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>18577</td>
</tr>
<tr>
<td>2</td>
<td>0.000000000</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>5432</td>
</tr>
<tr>
<td>3</td>
<td>0.000000000</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>18577</td>
</tr>
<tr>
<td>4</td>
<td>0.010749238</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>5432</td>
</tr>
<tr>
<td>5</td>
<td>0.010749238</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>5432</td>
</tr>
<tr>
<td>6</td>
<td>0.010749238</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>5432</td>
</tr>
<tr>
<td>7</td>
<td>0.010749238</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>5432</td>
</tr>
<tr>
<td>8</td>
<td>0.010749238</td>
<td>10.30.110.41</td>
<td>10.30.110.41</td>
<td>TCP</td>
<td>62</td>
<td>5432</td>
</tr>
</tbody>
</table>

Good to know if there is any additional header inside.
UDP Is Fast, But…

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Port</strong></td>
<td><strong>Destination Port</strong></td>
<td><strong>Length</strong></td>
<td><strong>Checksum</strong></td>
</tr>
</tbody>
</table>

Max datagram length is:
\[2^{16} - 20 \text{ (IPH)} - 8 \text{ (UDPH)} = 65,507 \text{ bytes}\]

Sep 19 01:35:01.452 m2m/mdt/backend-timer 0/RP0/CPU0 t15234 45924 [mdtbk_bte_encode_cb]: sub_id 5, /oper/optics/if/*/optics_info, len 77580: mdt_send_encoded_data returned error
To Encrypt or Not To Encrypt?

**gRPC Dial-in (NO-TLS)**

**Password exchange**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>00:01:45:05:62</td>
<td>10.38.110.41</td>
<td>10.38.110.40</td>
<td>HTTP2</td>
<td>79 WINDOW_UPDATE</td>
</tr>
<tr>
<td>11</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>992833 - 57960 [ACK] Seq=47 Ack=23 Win=3200</td>
</tr>
<tr>
<td>12</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>75 SETTINGS</td>
</tr>
<tr>
<td>13</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 43738 - 57960 [ACK] Seq=47 Ack=32 Win=3200</td>
</tr>
<tr>
<td>14</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>75 SETTINGS</td>
</tr>
<tr>
<td>15</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>225 HEADERS, DATA</td>
</tr>
<tr>
<td>16</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 43738 - 57960 [ACK] Seq=32 Ack=277 Win=4</td>
</tr>
<tr>
<td>17</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 57960 - 43738 [ACK] Seq=32 Ack=277 Win=4</td>
</tr>
<tr>
<td>18</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>281 HEADERS, DATA</td>
</tr>
<tr>
<td>19</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 43738 - 57960 [ACK] Seq=277 Ack=247 Win=4</td>
</tr>
</tbody>
</table>

**Message content**

```
6=LMed... .NC5501_topz.. test2...Cisco-1OS-XR-Shellutil-oper: system-time/uptime:
2015-01-07T. ...
H...P...ZB...z: keysz1: contentz: hostname: .NC5501_topz..
.uptime:8...
```

**gRPC Dial-in (TLS)**

**Password exchange**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>00:01:45:05:62</td>
<td>10.38.110.41</td>
<td>10.38.110.40</td>
<td>TCP</td>
<td>79 WINDOW_UPDATE</td>
</tr>
<tr>
<td>21</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>992833 - 57960 [ACK] Seq=47 Ack=23 Win=3200</td>
</tr>
<tr>
<td>22</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>75 SETTINGS</td>
</tr>
<tr>
<td>23</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 43738 - 57960 [ACK] Seq=47 Ack=32 Win=3200</td>
</tr>
<tr>
<td>24</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>75 SETTINGS</td>
</tr>
<tr>
<td>25</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>225 HEADERS, DATA</td>
</tr>
<tr>
<td>26</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 43738 - 57960 [ACK] Seq=32 Ack=277 Win=4</td>
</tr>
<tr>
<td>27</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 57960 - 43738 [ACK] Seq=32 Ack=277 Win=4</td>
</tr>
<tr>
<td>28</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>281 HEADERS, DATA</td>
</tr>
<tr>
<td>29</td>
<td>00:01:45:05:62</td>
<td>10.38.110.40</td>
<td>10.38.110.41</td>
<td>TCP</td>
<td>66 43738 - 57960 [ACK] Seq=277 Ack=247 Win=4</td>
</tr>
</tbody>
</table>

**Message content**

```
6=LMed... .NC5501_topz.. test2...Cisco-1OS-XR-Shellutil-oper: system-time/uptime:
2015-01-07T. ...
H...P...ZB...z: keysz1: contentz: hostname: .NC5501_topz..
.uptime:8...
```
Which Encoding To Use?

GPB: Message length: 330 bytes

KV-GPB: Message length: 1142 bytes

JSON: Message length: 1325 bytes
Design Your Transport Network Properly

Peak bandwidth consumption

- gRPC: 39.5 MBPS, 41.4 MBPS
- TCP: 39.7 MBPS, 41.7 MBPS
- UDP: 40.8 MBPS, 42.7 MBPS

315k counters
Every 5 seconds
Agenda

1. Brief Telemetry Overview
2. Closer Look at Router
3. Closer Look at the Link Between
4. Closer Look at Collector
5. Final Thoughts
Is Your Collector Fast Enough?

Make sure the collector has enough power to process your telemetry data.
Is Your Hard Drive Write Speed Fast Enough?

HDD-based server (SAS)

SSD-based server (SAS)
<table>
<thead>
<tr>
<th>Metric</th>
<th>InfluxDB</th>
<th>Prometheus</th>
<th>Elasticsearch</th>
<th>OpenTSDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Real-time Analytics</td>
<td>Monitoring System</td>
<td>Real-time Search</td>
<td>Real-time Analytics</td>
</tr>
<tr>
<td>Supported Measurements</td>
<td>metrics, events</td>
<td>metrics</td>
<td>metrics, events</td>
<td>metrics</td>
</tr>
<tr>
<td>High Availability (HA)</td>
<td>Double writing 2 servers</td>
<td>Double writing 2 servers</td>
<td>Clustering</td>
<td>Clustering</td>
</tr>
<tr>
<td>Underlying Technology</td>
<td>Golang</td>
<td>Golang</td>
<td>Java</td>
<td>Java, Hadoop</td>
</tr>
<tr>
<td>Storage Backend</td>
<td>Custom</td>
<td>Custom</td>
<td>Document</td>
<td>Hadoop (Columnar)</td>
</tr>
<tr>
<td>Supported Data Types</td>
<td>int64, float64, bool, and string</td>
<td>float64</td>
<td>string, int32, int64, float32,</td>
<td>int64, float32, float64</td>
</tr>
<tr>
<td>Bytes per point after compression</td>
<td>2.2</td>
<td>1.3</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>Metric Precision</td>
<td>nano second</td>
<td>milli second</td>
<td>milli second</td>
<td>milli second</td>
</tr>
<tr>
<td>Write Performance - Single Node</td>
<td>470k metrics / sec (custom HW)</td>
<td>800k metrics / sec</td>
<td>30k metrics / sec</td>
<td>32k metrics / sec (calculated)</td>
</tr>
<tr>
<td>Query Performance (1 host, 12hr by 1m)</td>
<td>3.78 ms (min), 8.17 (avg)</td>
<td>tbd</td>
<td>13.23 ms (min), 28.6 (avg)</td>
<td>tbd</td>
</tr>
<tr>
<td>Query Language</td>
<td>InfluxQL (SQL like)</td>
<td>PromQL</td>
<td>Query DSL</td>
<td>lookup only</td>
</tr>
<tr>
<td>Community Size</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td>medium</td>
</tr>
<tr>
<td>Maturity</td>
<td>Stable</td>
<td>Stable</td>
<td>Stable</td>
<td>stable</td>
</tr>
</tbody>
</table>

Full table: https://tinyurl.com/jsd4esy
Good to read: https://tinyurl.com/ybaw4ww6
Everything is working, where are my stats?

**RP/0/RP0/CPU0:ios-xr#** sh clock
Sun Apr 1 **20:56:15.074 PDT**
20:56:15.167 PDT Sun Apr 1 2018

cisco@ubuntu51:~$ date
Sun Apr 1 **23:13:11 PDT** 2018

---

**Subscription:** if-stats

---

**State:** ACTIVE

DSCP/QoS marked value: Default
Sensor groups:
Id: if-stats
  Sample Interval: 5000 ms
Sensor Path State: Resolved

Destination Groups:
Group Id: DGroup1
  Destination IP: 10.30.110.40
  Destination Port: 57500
  Encoding: self-describing-gpb
  Transport: grpc
  State: Active
Agenda

1. Brief Telemetry Overview
2. Closer Look at Router
3. Closer Look at the Link Between
4. Closer Look at Collector
5. Final Thoughts
Think About Your Overall Design
Think About Your Overall Design

Router

send_buffer

Collector 1

delays

Analytics/etc

Collector 2

window_update (slow down)

Analytics/etc
Think About Your Overall Design

Router

send_buffer

Collector 1

Analytics/etc

Collector 2 (delays)

Analytics/etc
Think About Your Overall Design

Router → send_buffer → Collector 1
(Analytics/other)
(delays)
Collector 2
(designs)
window_update (speed up) → Analytics/other
Think About Your Overall Design

What do you want your router to send now?

Do you want your router to slow down here instead?

Or do you want your router to go through collections twice?

Recent data

Old data

send_buffer

Router

Collector 1

Analytics/etc

Collector 2 (delays)

Analytics/etc
It Is Good to Know More

A snapshot from https://www.youtube.com/watch?v=ZMZJ3ZaEeIQ