DEEP DIVE IN ZABBIX PRE-PROCESSING

Arturs Lontons
ZABBIX Technical Support Engineer
WHY PREPROCESSING?

The retrieved data is freeform and is not fit for calculations, aggregations and/or optimal data storage:

<table>
<thead>
<tr>
<th>Uptime: 184000</th>
<th>Threads: 19</th>
<th>Questions: 37</th>
</tr>
</thead>
<tbody>
<tr>
<td>10986</td>
<td>Slow queries: 0</td>
<td>Opens: 101</td>
</tr>
<tr>
<td></td>
<td>Flush tables: 2</td>
<td>Open tables: 127</td>
</tr>
<tr>
<td></td>
<td>Queries per second avg: 20.168</td>
<td></td>
</tr>
</tbody>
</table>
## WHY PREPROCESSING?

## HOW CAN I SOLVE THIS?

**USE PREPROCESSING!**

<table>
<thead>
<tr>
<th>Text preprocessing</th>
<th>Deltas</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured data</td>
<td>Numeral systems</td>
<td>Prometheus Exporter</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>Javascript</td>
<td></td>
</tr>
</tbody>
</table>
THE EVOLUTION OF PREPROCESSING

DEEP DIVE IN
ZABBIX PRE-PROCESSING
THE LEGACY WAY

Limited preprocessing support in versions <3.4:

- Custom multiplier
- Numeral system transformations
- Delta calculation (speed per second/simple change)

What if you needed more?

Preprocess the data by using your own scripts!
VERSIONS 3.4 AND 4.2

What has changed?

ZABBIX VERSION 3.4

Introducing - Preprocessing!
New ways to transform data:
• Regex
• Trim
• XML XPath
• JSON Path

ZABBIX VERSION 4.2

• Extended preprocessing
• Validation and throttling
• Custom error handling
• Preprocessing support by Zabbix Proxy
• Prometheus exporter support
VERSION 4.4

What has changed?

- Preprocessing of XML data via Xpath
- JSONPath aggregation and search
- Extended Custom error handling
- Introducing CSV to JSON preprocessing
- WMI, JMX and ODBC data collection returns JSON arrays – ready to preprocess via JSONPath!
THE MANY WAYS OF PREPROCESSING

DEEP DIVE IN
ZABBIX PRE-PROCESSING
TEXT PREPROCESSING

Retrieve a value by using Regex:

Temperature: 36C → PCRE → 36

Trim the retrieved value and store it as a number:

36C → Right Trim → 36
Retrieve value from JSON or XML data:

```xml
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>

  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

- $\text{sum}(/\text{bookstore/book/price}) = 99.94$
- $\text{count}(/\text{bookstore/book}) = 3$
- $\text{number}(/\text{bookstore/book[price<30]/price}) = 29.99$
ARITHMETIC

Custom Multipliers to transform numeric data:

<table>
<thead>
<tr>
<th>Preprocessing steps</th>
<th>Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Custom multiplier</td>
<td>0.125</td>
</tr>
</tbody>
</table>

DELTA CALCULATIONS

- Difference between current and previous value
- Change per second
INTRODUCED IN VERSION 4.2

- Custom scripts
- Validation
- Throttling
- Prometheus

**Custom scripts**
- JavaScript

**Validation**
- In range
- Matches regular expression
- Does not match regular expression
- Check for error in JSON
- Check for error in XML
- Check for error using regular expression

**Throttling**
- Discard unchanged
- Discard unchanged with heartbeat

**Prometheus**
- Prometheus pattern
- Prometheus to JSON
JAVASCRIPT

Implemented with **Duktape** JavaScript engine!

With JavaScript you can perform:

- Data transformation
- Data aggregation
- Data filtering
- Logical expressions
- etc.

...All done internally by Zabbix
Convert **diskstats** to JSON:

```javascript
function (value) {
  var parsed = value.split("\n").reduce(function(acc, x, i) {
    acc["values"][x.split(/ +/)[3]] = x.split(/ +/).slice(1);
    acc["lld"].push({"#{DEVNAME}" : x.split(/ +/)[3]});
    return acc;
  }, {"values" : {}, "lld" : []});

  return JSON.stringify(parsed);
}
```
JAVASCRIPT

Initial data

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 8 | 0 | sda | 9224 | 3 | 606559 | 8510 | 207906 | 1416 | 7707716 | 275248 | 0 | 219108 | 283619 |
| 8 | 1 | sda1 | 969 | 0 | 12390 | 146 | 10 | 0 | 4136 | 14 | 0 | 153 | 160 |
| 8 | 2 | sda2 | 8230 | 3 | 590897 | 8352 | 157268 | 1416 | 7703580 | 98777 | 0 | 48122 | 106992 |
| 11 | 0 | sr0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 253 | 0 | dm-0 | 8061 | 0 | 581737 | 8311 | 203596 | 0 | 7703556 | 292885 | 0 | 219283 | 301196 |
| 253 | 1 | dm-1 | 90 | 0 | 4920 | 32 | 3 | 0 | 24 | 5 | 0 | 21 | 37 |

Preprocessed data - ready for LLD!

```
{"values":{"sda":
["8","0","sda","9248","3","607351","8535","219127","1432","8108169","286348","0","228233","294732"],"sda1":
["8","1","sda1","969","0","12390","146","10","0","4136","14","0","153","160"],"sda2":
["8","2","sda2","8248","3","591617","8371","165831","1432","8104033","102626","0","50209","110848"],"sr0":
["11","0","sr0","0","0","0","0","0","0","0","0","0","0","0"],"dm-0":["253","0","dm-0","8079","0","582457","8330","214576","0","8104009","304043","0","228416","312373"],"dm-1":["253","1","dm-1","90","0","4920","32","3","0","24","5","0","21","37"],"lld":{"{"DEVNAME":"sda"},{"DEVNAME":"sda1"},{
"DEVNAME":"sda2"},{"DEVNAME":"sr0"},{"DEVNAME":"dm-0"},{"DEVNAME":"dm-1"}}}
```
Result:

<table>
<thead>
<tr>
<th>disk stats: diskstats: sr0 Disk read rate</th>
<th>vfs.dev.read.rate[sr0]</th>
<th>90d</th>
<th>365d</th>
<th>Dependent item</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk stats: diskstats: sda Disk read rate</td>
<td>vfs.dev.read.rate[sda]</td>
<td>90d</td>
<td>365d</td>
<td>Dependent item</td>
<td>Enabled</td>
</tr>
<tr>
<td>disk stats: diskstats: sda2 Disk read rate</td>
<td>vfs.dev.read.rate[sda2]</td>
<td>90d</td>
<td>365d</td>
<td>Dependent item</td>
<td>Enabled</td>
</tr>
<tr>
<td>disk stats: diskstats: sda1 Disk read rate</td>
<td>vfs.dev.read.rate[sda1]</td>
<td>90d</td>
<td>365d</td>
<td>Dependent item</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

The items have been discovered and created by our LLD!
VALIDATION

Validate data against validation logic:

- In range
- Matches regular expression
- Does not match regular expression
- Check for errors in JSON/XML or by using regex

CUSTOM ON FAIL

Define the behavior in cases when the preprocessing fails:

- Discard value
- Set value to
- Set error to
VALIDATION

Discard the value outside of defined range:

Match a regular expression and set the value to Unknown if no match is obtained:
VALIDATION

Check for an application-level error message located at JSONPath/XPath:

Preprocessing steps | Name | Parameters
--- | --- | ---
1: Check for error in JSON | $.Application.ErrorMessage

```json
{
    "Software": "Zabbix",
    "Version": "4.2.0",
    "OS": "CentOS 7",
    "ErrorMessage": "Service Down"
}
```

Error message!
THROTTLING

Enables high frequency monitoring with minimal performance impact:

- Discard repeating values
- Discard repeating values with a heartbeat

Useful when we are receiving a lot of duplicate data at a very high frequency!

<table>
<thead>
<tr>
<th>Application service monitoring:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time:</strong></td>
</tr>
<tr>
<td><strong>Data:</strong></td>
</tr>
</tbody>
</table>
WITH THROTTLING

Zabbix Server → History

0 1 0 0 0 0 1 0 1 0 1 1 0
THROTTLING WITH A HEARTBEAT

Zabbix Server

History

0 1 0 0
0
0
0
0
0
1
1
0

0 1
0
-
-
0 (Heartbeat)
1
-
0
PROMETHEUS

- Query Prometheus endpoint with HTTP checks
- Use preprocessing to obtain metrics
- Use within LLD to discover components monitored by Prometheus

Databases
- Aerospike exporter
- ClickHouse exporter
- Consul exporter (official)
- Couchbase exporter
- CouchDB exporter
- ElasticSearch exporter
- EventStore exporter
- Memcached exporter (official)
- MongoDB exporter
- MSSQL server exporter
- MySQL router exporter
- MySQL server exporter (official)

Issue trackers and continuous integration
- Bamboo exporter
- Bitbucket exporter
- Confluence exporter
- Jenkins exporter
- JIRA exporter

HTTP
- Apache exporter
- HAProxy exporter (official)
- Nginx metric library
- Nginx VTS exporter
- Passenger exporter
- Squid exporter
- Tinyproxy exporter
- Varnish exporter
- WebDriver exporter

...And many more
PROMETHEUS PATTERN

- Create master HTTP item
- Create dependent items with “Prometheus pattern”

Retrieve a metric:

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameters</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prometheus pattern</td>
<td>▼ node_load1</td>
<td>&lt;label name&gt;</td>
</tr>
</tbody>
</table>

Retrieve a label value:

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameters</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prometheus pattern</td>
<td>▼ node_network_speed_bytes</td>
<td>device</td>
</tr>
</tbody>
</table>
PROMETHEUS LLD

Create a dependent item LLD with a “Prometheus to JSON” preprocessing step.

Discover all CPU’s:

<table>
<thead>
<tr>
<th>Preprocessing steps</th>
<th>Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prometheus to JSON</td>
<td>node_cpu_seconds_total{cpu=&quot;,+&quot;,mode=&quot;,+&quot;}</td>
</tr>
</tbody>
</table>
PROMETHEUS LLD

Retrieved JSON:

```json
[
  {
    "name": "node_cpu_seconds_total",
    "value": "88798.31",
    "line_raw": "node_cpu_seconds_total{cpu="0",mode="idle"} 88798.31",
    "labels": {
      "cpu": "0",
      "mode": "idle"
    },
    "type": "counter",
    "help": "Seconds the cpus spent in each mode."
  }
]
```
PROMETHEUS LLD

Retrieved JSON:

```json
[
  {
    "name": "node_cpu_seconds_total",
    "value": "88798.31",
    "line_raw": "node_cpu_seconds_total{cpu="0",mode="idle"} 88798.31",
    "labels": {
      "cpu": "0",
      "mode": "idle"
    },
    "type": "counter",
    "help": "Seconds the cpus spent in each mode."
  }
]
```

Macros: `{#CPUNUM}`, `{#MODE}`

JSONPath: `$.[0].labels.cpu`, `$.[0].labels.mode`

Macro: `{#HELP}`

JSONPath: `$.[0].help`
PROMETHEUS LLD

Item prototype with “Prometheus pattern” preprocessing step:

```
node_cpu_seconds_total{cpu="{#CPUNUM}",mode="{#MODE}"}
```
Use `{#HELP}` to populate the Description field:

```
"help":"Seconds the cpus spent in each mode."
```

Macro: `{#HELP}`

JSONPath: `$.help`
<table>
<thead>
<tr>
<th>Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE</th>
<th>seconds_total_[0, user]</th>
<th>90d</th>
<th>Dependent item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE system</td>
<td>seconds_total_[0, system]</td>
<td>90d</td>
<td>Dependent item</td>
</tr>
<tr>
<td>Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE steal</td>
<td>seconds_total_[0, steal]</td>
<td>90d</td>
<td>Dependent item</td>
</tr>
<tr>
<td>Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE softirq</td>
<td>seconds_total_[0, softirq]</td>
<td>90d</td>
<td>Dependent item</td>
</tr>
<tr>
<td>Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE nice</td>
<td>seconds_total_[0, nice]</td>
<td>90d</td>
<td>Dependent item</td>
</tr>
<tr>
<td>Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE irq</td>
<td>seconds_total_[0, irq]</td>
<td>90d</td>
<td>Dependent item</td>
</tr>
<tr>
<td>Metric</td>
<td>Time Window</td>
<td>Dependent</td>
<td>Last Check</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>seconds_total_[0, idle]</td>
<td>90d</td>
<td>Depend...</td>
<td>2019-10-02 16:5...</td>
</tr>
<tr>
<td>seconds_total_[0, iowait]</td>
<td>90d</td>
<td>Depend...</td>
<td>2019-10-02 16:5...</td>
</tr>
<tr>
<td>seconds_total_[0, irq]</td>
<td>90d</td>
<td>Depend...</td>
<td>2019-10-02 16:5...</td>
</tr>
<tr>
<td>seconds_total_[0, nice]</td>
<td>90d</td>
<td>Depend...</td>
<td>2019-10-02 16:5...</td>
</tr>
<tr>
<td>seconds_total_[0, softirq]</td>
<td>90d</td>
<td>Depend...</td>
<td>2019-10-02 16:5...</td>
</tr>
<tr>
<td>seconds_total_[0, steal]</td>
<td>90d</td>
<td>Depend...</td>
<td>2019-10-02 16:5...</td>
</tr>
<tr>
<td>seconds_total_[0, system]</td>
<td>90d</td>
<td>Depend...</td>
<td>2019-10-02 16:5...</td>
</tr>
</tbody>
</table>
All of the preprocessing is being done by the server!
PREPROCESSING WITH VERSION $\geq 4.2$

Preprocessing is performed on the proxy!

Raw data $\rightarrow$ Zabbix Proxy $\rightarrow$ Zabbix Server $\rightarrow$ Preprocessed data

Compressed preprocessed data $\rightarrow$ History
PREPROCESSING UNDER THE HOOD

DEEP DIVE IN
ZABBIX PRE-PROCESSING
PREPROCESSING WORKFLOW

Data source

Preprocessing manager

Enqueue item value

Send value to worker

Add result to queue

Value stored

Preprocessing worker

Execute preprocessing steps
PREPROCESSING MANAGER

● Added in version 3.4
● Enqueues the items in the preprocessing queue
● Assigns the preprocessing tasks to preprocessing workers
● Flushes the preprocessed values from the queue

PREPROCESSING WORKERS

● StartPreprocessors defines the value of pre-forked preprocessing workers
● Number of workers determined by:
  ○ Count of preprocessable items
  ○ Count of preprocessing steps
  ○ etc.
LET’S RECAP

Automate!
- Use master items with LLD!
- Discover your metrics with preprocessing!

Improve!
- Discard unnecessary data with throttling!
- Improve performance by preprocessing on the proxy!

Customize!
- Data validation!
- Custom behavior with advanced preprocessing rules!

Transform!
- Transform your data!
- Enable data aggregation and calculation with preprocessing!
THANK YOU!

Arturs Lontons
ZABBIX Technical Support Engineer