|  |  |
| --- | --- |
| **Component** | C plugin for Collectd (Version 5.5.0) |
| **File name** | **Monitoring\_Collectd\_On\_The\_Fly\_Aggregator.docx** |
| **Use case(s)** |  |
| **Requirement(s)** | S10 |
| **Quality attribute(s)** | 5.4.3.3 Customizability |
| **Revision history** | 07/11/2015 – Created by Joel Gao  07/12/2015 – Modified by Joel Gao |

**Component Responsibilities (functions):**

The on-the-fly aggregator is a plugin for Collectd. It allows to aggregate multiple values into a single value using one or more aggregation functions, e.g. sum, average. This has a broad range of applications, e.g. calculating the average CPU utilization over all cores of each host. It should be deployed alone with the Collectd program on server node. Its responsibilities include:

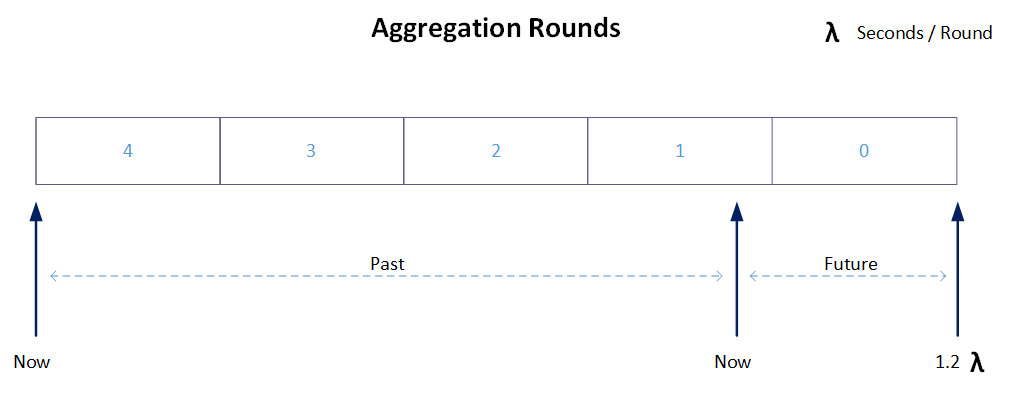
* Read configuration parameters
* Cache metric values for fault tolerance
* Calculate aggregation values based on raw data using different functions
* Submit values

**Relationships with other components:**

**Collectd server**: this component is a plugin of collectd.

**Major Design Decisions:**

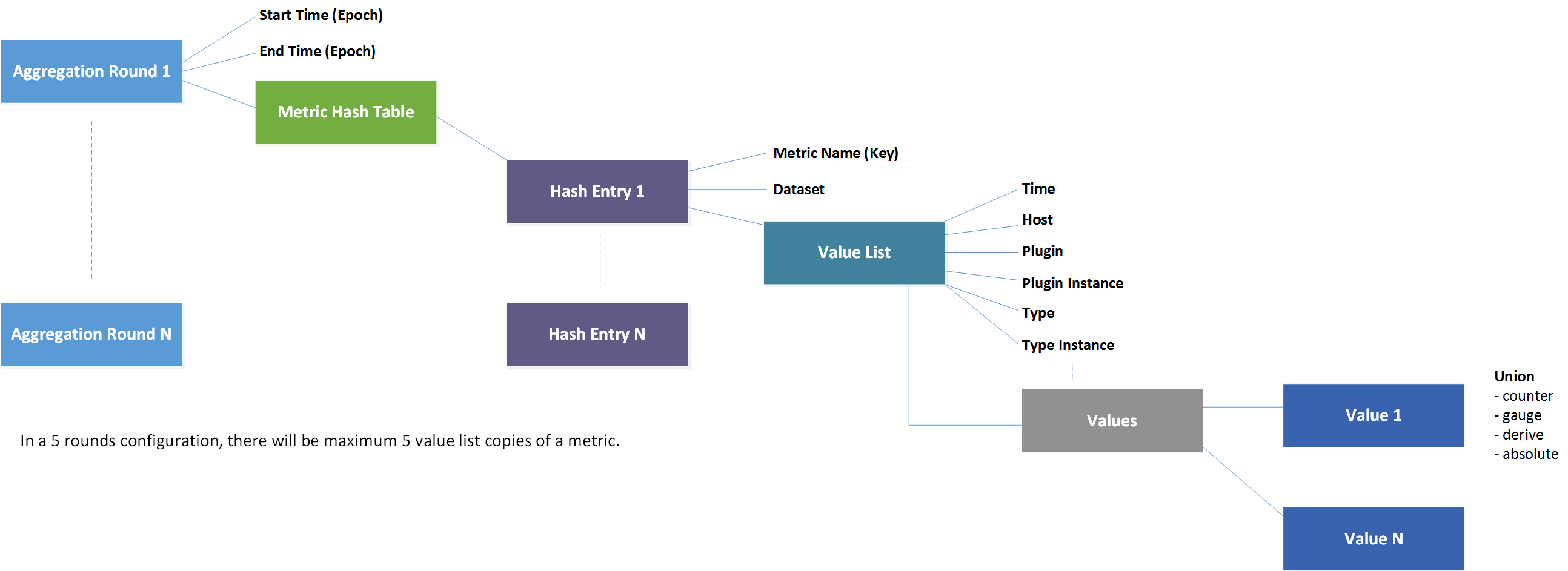
1. To achieve functionalities and performance, C is used to create this program
2. Multiple rounds aggregation is designed for reliability
3. Hash table is used to cache metric values inside the new plugin. Metric path is the string key (host + plugin + plugin instance + type + type instance). An aggregation round structure was created. It has start and end time of this round and the hash table for values. If there are multiple values with same key arrive in the same round, the last value will be used. Metric collected time (not metric received time) is used for aggregation. This means that even if the metrics arrive early or late, they can always be aggregated in the correct round. Now the problem is how many rounds to key. This is an important parameter. If it's very large, we may need more memory to cache the data (depending on the aggregation configuration), the time will be longer to show aggregated data on dashboard, but the fault tolerance will be better. If the nodes time is not well synchronized or the network is flaky, more rounds can help to improve correctness. This is a **tradeoff between resource utilization, performance (response time), and reliability**. The default number is 5, you can change the value of AGG\_RETENTION\_ROUND (and recompile, I need to change this to be configurable) to fine tune for your system.
4. Aggregation round



As shown above, this is a 5-round configuration. At any time, round 0 is for future metrics. This is useful if time of other nodes is faster than server node. Round 1 will receive current data. Round 2, 3, and 4 are for past data. Given the behavior of network plugin (see justification), metric data is more likely to arrive for round 1 – 4.

1. Data structure design

Diagram below shows the main data structure used for aggregation.



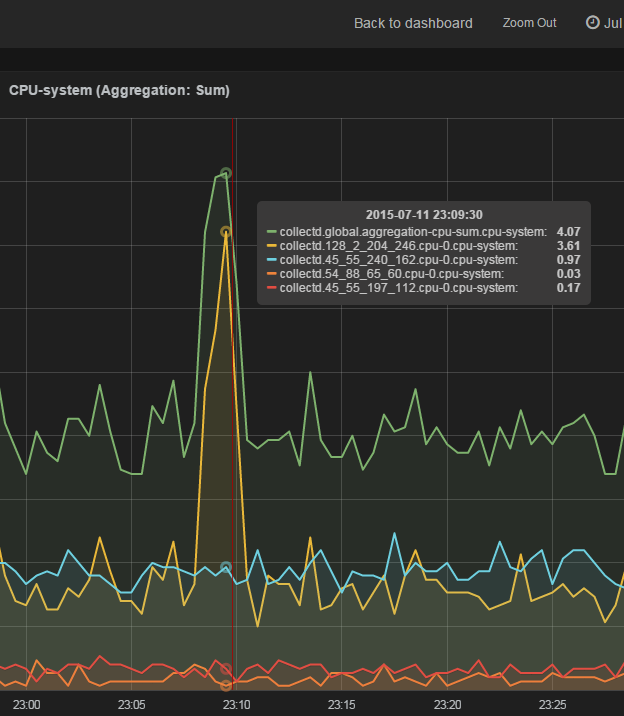
**Justification:**

Collectd 5.5.0 Aggregation problem

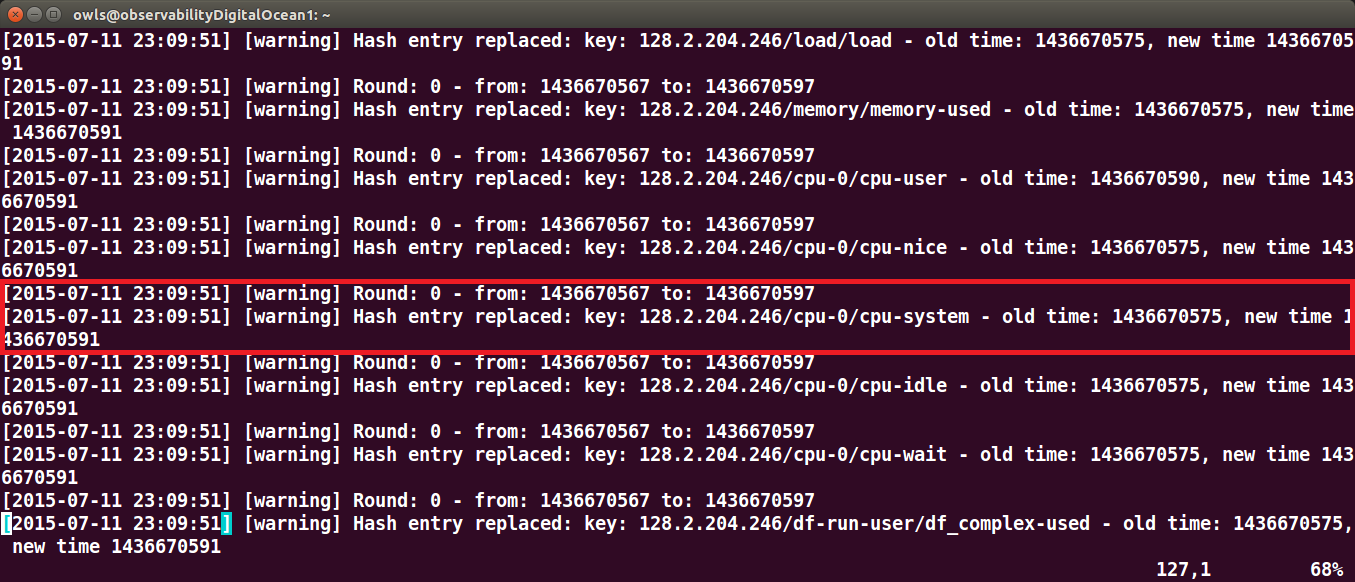
Current aggregation plugin works well on a single machine to aggregate local metrics values. However, if you want to use it on a Collectd server to aggregate metrics across multiple machines, you won't get correct results. The main reason for this is that the network plugin is not designed to transmit real-time metric values to server. Suppose you are aggregating CPU-system metric in a 30 seconds interval for a cluster with node A, B, C, D. The Collectd server is configured on F. In the latest 30 seconds, F is not likely to get metrics data it needs from A, B, C, and D. This is because the network plugin tries to cache outgoing data and send it in chunk. Thus, it is the normal case that for some aggregation round, there is no value from node A. But for the next round, there are multiple values arrive at the same time. One simple solution to this is to modify the network plugin to let it flush every metric value in each read call back. However, in a large system with thousands of nodes, this will increase network traffic and add more overhead on server node. Current design tries to send packets in size of MTU, which is good for performance consideration. Another solution is to modify the aggregation plugin so that it can aggregate correctly (as described in this document).

Another problem is that in existing aggregation plugin, it automatically gets latest cached value as rate to aggregate (Collectd uses AVL tree to cache metric data. (Search time is O(longN)).). This will give wrong result for aggregation of historical data. Why using rate, because in aggregation we can specify up to 6 different methods (sum, average...) and the metric value can be gauge, counter, derive, and absolute. Converting to rate will simply the calculation. In new aggregation plugin described in this document, a hash table is used to store the value to rate states so that each value can be converted correctly.

**Known Issue:**



As shown in the diagram above, the aggregated **sum** value (green) is not correct. This was the only error in 24 hours. After reading the log of aggregator, I found the following warning.



In a 30 seconds collecting interval, server received two values for cpu-0/cpu-system from same node. The previous value was replaced. Which generated the false result.

One possible reason could be the cpu was suddenly heavily used (as shown in graph), Collectd’s threads could not get executed in time. The cpu plugin thread was scheduled later and fell into the same round as next collecting process. Thus two values were generated in one round.

**Extra Documentation (e.g. Java Doc):**

Sample aggregation configuration:

LoadPlugin "aggregation"

<Plugin "aggregation">

<Aggregation>

Plugin "cpu"

Type "cpu"

GroupBy "TypeInstance"

CalculateNum false

CalculateSum true

CalculateAverage true

CalculateMinimum false

CalculateMaximum false

CalculateStddev true

</Aggregation>

</Plugin>