The Gatherer - a mechanism for integration of monitoring data in ATLAS

Peter Renkel[1], **Haleh Hadavand**[1], **Robert Kehoe**[1]

[1] Physics Department, Southern Methodist University, Dallas, Texas

E-mail: renkel@fnal.gov

Abstract. The ATLAS experiment's triggering system is distributed across large farms of approximately 1500 nodes. Online monitoring and data quality runs alongside this system. We have designed a mechanism that integrates the monitoring data from different nodes and makes it available for shift crews. This integration includes adding or averaging of histograms, summing of trigger rates, and combination of more complex data types across multiple monitoring processes. Performance milestones have been achieved which ensure the needs of early datataking will be met. We will present a detailed description of the architectural features and performance.

1. ATLAS Online Monitoring

ATLAS is one of the four experiments along the Large Hadron Collider (LHC) ring at CERN (European Organization for Nuclear Research). It is intended to collide two beams of either protons traveling in opposite directions at a center-of-mass of 14 TeV at a rate of 40MHz or lead ions. Due to the enormous rate of incoming information in ATLAS detector, the data should be split among a number of monitoring nodes. The monitoring algorithms running in each trigger node are called 'providers'. Digitized data from all sub-detectors flows through Level 2 and subsequent Event Filter trigger farms of ∼1500 nodes each. Partial event fragments are available at Level 2, and fully assembled events are available in Event Filter. Monitoring algorithms running within these farms generate histograms or other results which are stored in online histogram servers [3] (OH) or information service servers [2] (IS), respectively. Figure 1 shows the schematic of the online monitoring framework.

Each provider within a processing farm receives just a fraction of all events. An online mechanism that integrates data, from the different monitoring nodes, called 'the Gatherer' is shown on top of Fig. 1. The Gatherer has access to all results of a specific type from various providers. It sums or otherwise combines those results, and retransmits those results back to IS/OH servers, where it can be accessed by shifter crews.

2. The Gatherer

The results from different providers are regularly sent [1] to the IS or OH servers [3]. The period between two updates may vary but is typically chosen to be around 1 min. The Gatherer is aimed at receiving the information from these servers, collecting it and publishing back to the servers. This includes mainly summation or averaging of histograms, but can include summation of trigger rates and more complicated objects.

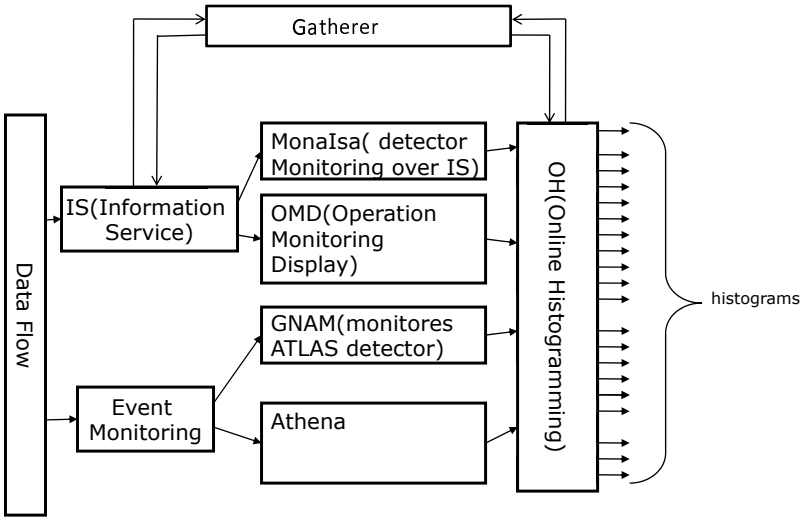

Figure 1. Online Monitoring Framework. Gatherer (top) integrates data coming from different monitoring nodes on IS/OH servers, sends it back to IS/OH to be monitored by shifter crews

The number of monitoring nodes in year 2010 can be as high as 1500, and we expect 100 histograms per node. This means that the Gatherer should be able to process 150,000 histograms in the periods of time between two consecutive updates (∼1min). Therefore, it is extremely important to optimize this component as much as possible utilizing the most efficient summation algorithms and taking advantage of multithreading technologies (Section 3.1.2).

2.1. Tier structure

One technique to increase performance includes splitting of the gathering mechanism into several tiers. Fig. 2 illustrates this idea. Each sub-farm of the monitoring nodes in this configuration will be associated with individual Gatherer applications. In such a configuration, each of these Gatherers will collect and publish sums for the histograms coming from the corresponding sub-farm. A top-level Gatherer will then collect the published sums from the individual Gatherers, sum them, and publish the final results of summation. The above mentioned configuration can be split again into summation starting from sub-sub-farms to sub-farms. Each Gatherer collects just a portion of all monitoring data, and the load is split among the Gatherers, thus decreasing CPU and memory consumption per Gatherer. This mechanism also optimizes the time needed to collect the data since there are several Gatherers per layer working synchronously. Typically, there are two tiers with ∼10 providers per Gatherer. This number will be increased to $sim40$ providers per Gatherer in 2010.

2.2. Naming conventions

The necessity to differentiate between different histograms and between histograms coming from different monitoring nodes requires to develop a naming convention, unique for all histograms, and understandable by the Gatherer and other monitoring applications. The template for the general histogram name is as follows

$$< server \quad name > . < provider \quad name > . < histogram \quad name >,$$

where the $< server \quad name >$ stands for the name of the server where a given histogram is stored, the $< histogram \quad name >$ is the actual name of a histogram (e.g. "LepPt" for the transverse momentum of leptons), and $< provider \quad name >$ corresponds to the name of a

provider or node. The Gatherer collects and sums histograms with the same histogram and server name, coming from different providers. The specific instructions to the Gatherer are given by regular expressions [4]. For example, setting the parameter 'Provider' to 'L2PU-.*' in the configuration instructs the Gatherer to sum all histograms coming from Level 2 providers. Analogously, one can ask the Gatherer to sum across many IS/OH servers.

3. Summation architecture

In order to perform the gathering operation, the Gatherer must access the results, combine them to a final result, and publish them. Several different types of result must be accommodated, with implications for the type of gathering operation performed. Given the large number of results to be gathered, several choices must be made which guarantee the fastest operation.

3.1. Flow of results

3.1.1. Collecting data In order to collect the data from IS/OH Servers, we designed a *pull* approach to acquire result updates. Figure 3 shows the comparison between *push* and *pull* approaches. *Push* approach scans for data in the IS/OH Servers or repositories, and collects them at the same time, thus involving network traffic. The *pull* approach is much faster and does not involve extensive network communications. The Gatherer subscribes to a specific sort of information through the mechanism available in [2]. If a histogram is updated on the OH server, a notification is sent to the Gatherer. The Gatherer, in turn, applies the correspondent summation algorithm to this information and, if needed, publishes the result. This ensures that the information is summed as fast as it is updated (∼100 times faster than with the *push* method).

3.1.2. Multithreading To further optimize the performance, the summation processes that do not share the same resources run in different threads thus optimizing the time needed for summation. One example of processes that don't share the same resources is summing histograms with different names.

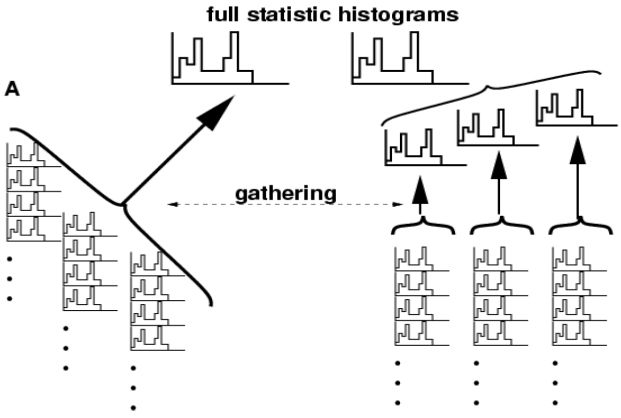

Figure 2. Possible gathering policies. Single Gatherer mode (left) where all data is gathered by one process and transmitted to IS/OH. Tired Gatherer mode (right) where the gatherer mechanism is split into several tiers.

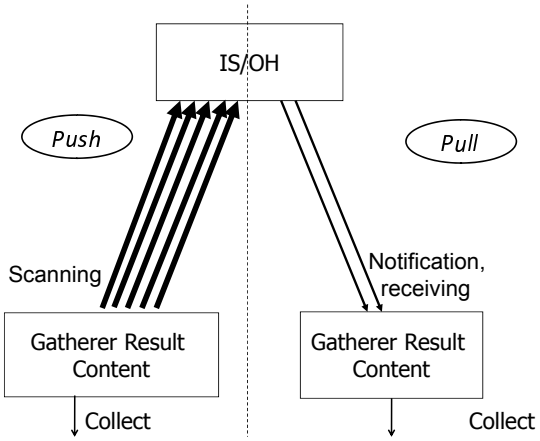

Figure 3. Comparison of Push and Pull methods for summation of histograms. In Push mode (left), the Gatherer scans for information in IS/OH servers involving substantial network traffic, while in Pull Mode (right), it subscribes for the information, and gets notification when new data flows in, or when data on server is updated.

3.1.3. Publishing The Gatherer stores the number of providers per a histogram name. Each time when the current number of updated providers reaches this number, a corresponding sum is published. The number of providers per a histogram names is updated whenever one or several monitoring nodes don't respond any longer or start responding.

3.2. Gatherer Operations

3.2.1. Histograms The Gatherer collects and sums one, two and three dimensional histograms, profile histograms, and graphs. Presently, three summation algorithms are implemented: adding, averaging, and adding with normalization of result. When the histograms to be summed have different binning or distinct axes limits, an attempt is made to rebin such histograms so they can be added. This feature can be turned off if it is known that the histograms are compatible with each other.

A special attention is given to the labeled histograms. This means that if histograms have alphanumeric labels instead of numbers, special summation algorithms should be implemented. These algorithms should sum bins with identical labels. This means that if the order of alphanumeric labels in two histograms differ, the labels in the second histogram are rearranged to match those in the first one. When there are more labels in the second histogram compared to the first one, the new labels are inserted after the last label of the first histogram.

For example, when a histogram with alpha-numeric labels $Voltage_1$, $Voltage_2$, $Voltage_3$ with corresponding values of 1.0,2.0,3.0 is to be summed with a histogram with alpha-numeric labels $Voltage_2$, $Voltage_4$ and corresponding values of 1.0, 2.0, the resulting histogram will contain alphanumeric labels $Voltage_1$, $Voltage_2$, $Voltage_3$,$Voltage_4$ with the corresponding values of 1.0,3.0,3.0,2.0.

3.2.2. Different modes of gathering As it was mentioned before, the Gatherer is able to sum not only histograms, but much more complex objects, e.g. numbers, array of numbers, boolean variables, strings and collection of these objects. In each case a special algorithm for summation of specific objects is applied. Figure 4 shows the summation of collections of a boolean, a number, a 1-dimensional histogram, an array of integers and floats.

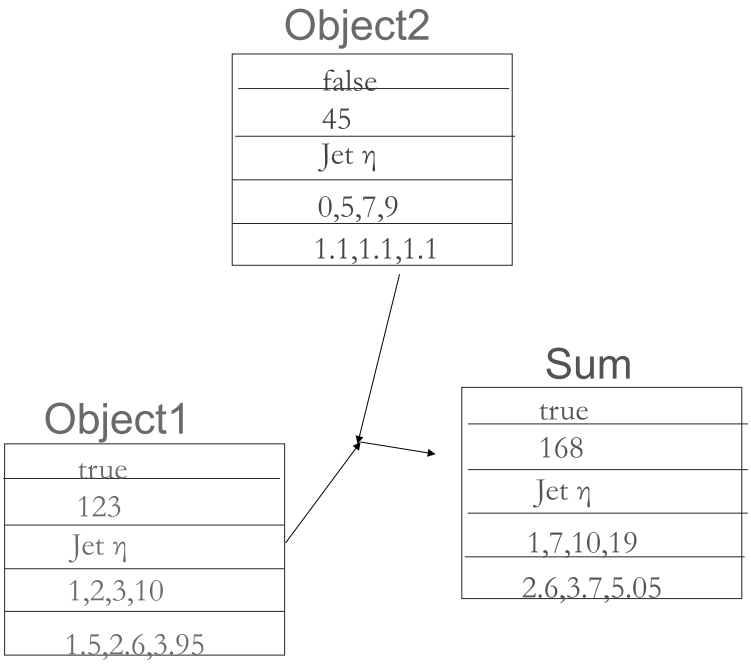

Figure 4. Summation of complex objects. Each field of a complex object is summed independently of all others.

3.2.3. Detector/Software problems testing Often, it is needed to trace the evolution of a specific issue of the detector or software. This feature is implemented in the Trigger Data Acquisition System by considering many versions of a histogram, each with specific history tag. Each history tag corresponds to the time when a histogram was updated. The Gatherer in this case will sum histograms with the same tag. Each gathered histogram can then be presented in an array such that the time history of the particular result is visible. The Gatherer can also be configured to sum just N latest tags of a histogram and ignore all others. Fig. 5 illustrates this idea.

3.2.4. Gatherer Configuration There are two major blocks in the Gatherer configuration: the HistogramReceiverConfig and HistogramProviderConfig. The information on the objects to be gathered (regular expressions for the histograms and provider names, number of history tags to consider, and OH servers where the histograms are located) is given in the HistogramReceiverConfig. The information on the gathered sums and modes of gathering (name of provider with which the sums are published, information on how to merge the histograms and what summing algorithm to utilize, and the OH server where the summed histograms are published) is given in the HistogramProviderConfig. A relevant piece of the Gatherer configuration is shown below:

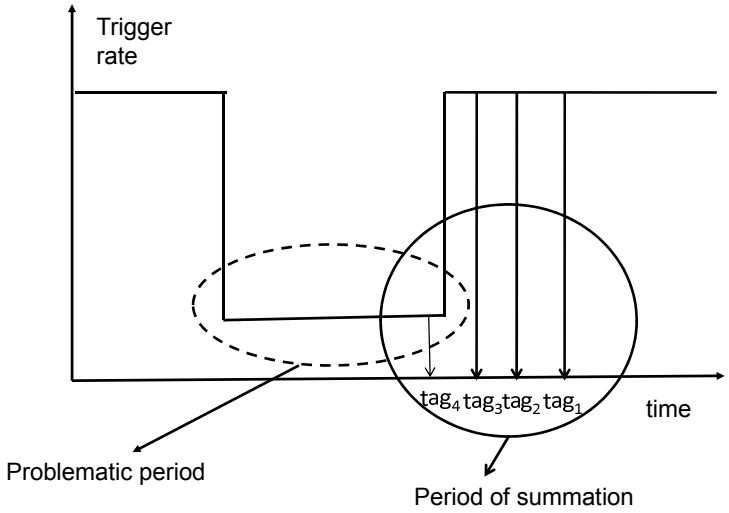

Figure 5. Trigger rate as a function of time. Trigger rate is clearly suppressed in the middle of the diagram (dashed oval). The solid circle represents the trigger rates at five different instances of time summed by the Gatherer. The Gatherer will monitor one suppressed trigger rate at time labeled tag_4 and three normal trigger rates (tag_3,tag_2,tag_1). Therefore, it will be able to report a drop of trigger rate labeled tag_4.

```
< obj class="HistogramProviderConfig" id="Gatherer-Provider-L2-1">
    < attr name="PublishProviderName" type="string">"L2-H"< /attr >
    < attr name="SummingMode" type="enum">"AdvancedMerge"< /attr >
    < attr name="ResultContent" type="enum">"Sum"< /attr >
    < rel name="Storage" num="1">
        "HistogramStorage" "Common-Histogram-Storage"
    < /rel >
</obj>
< obj class="HistogramReceiverConfig" id="Gatherer-Receiver-L2-1">
    < attr name="Providers" type="string">"(L2PU—L3PU).*"< /attr >
    < attr name="Histograms" type="string">".*"< /attr >
    < attr name="SumTags" type="u32">4< /attr >
    < rel name="Storage" num="2">
        "HistogramStorage" "Gatherer-Receiver-L21"
        "HistogramStorage" "Gatherer-Receiver-L22"
    < /rel >
< /obj >
```

In the above example, each attribute of the configuration is given by the three parameters: the name of the attribute, its type ("string","enum","u32", where "u32" stands for some form of integer), and its value. Their can be several OH servers from which the histograms are taken, so the "Storage" attribute is split into several fields, each field specifying an OH server.

3.3. Internal Synchronization

3.3.1. Command forwarding In case when we want to quickly diagnose the monitoring system (between two updates), a method to inquiry of OH server is needed. This method is implemented as follows. A signal (e.g. to publish) is sent to the top-level Gatherer. This signal is then

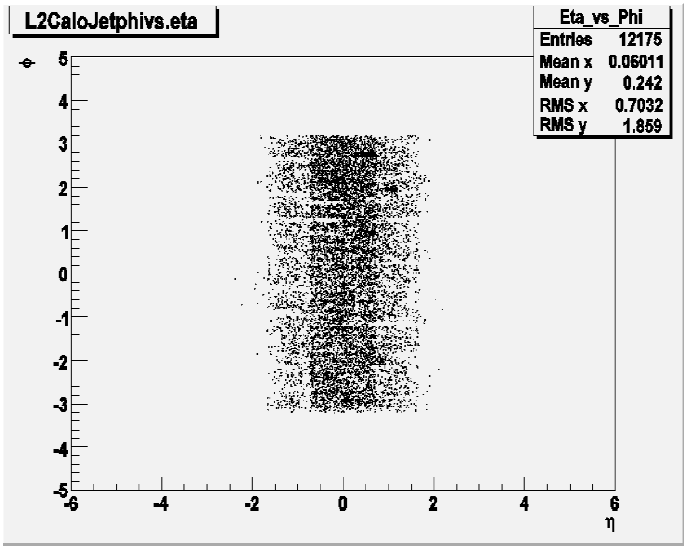

Figure 6. An example of the summed histogram of jet ϕ and η. Sum includes 8 providers.

transmitted to the lower tiers, the Gatherers sends it to the histograms, the histograms are published, notification are sent to the Gatherers (*pull* mode) and the sums are produced.

3.3.2. Exiting After a run is ended, a notification is sent to the monitoring applications. If the Gatherer is split in several tiers, it is necessary to synchronize those tiers. The top-level Gatherer will exit only after receiving the histograms from lower tiers. The implementation is illustrated with the following scheme.

- A run is ended and a notification is sent;
- The Gatherers of the first tier collect histograms, publish sums, and exit;
- The Gatherers of the second tier collect histograms, publish sums, and exit;
- ...
- The Gatherers of the k^{th} tier collect histograms, publish sums, and exit;
- ...
- The top-level Gatherer collects histograms, publish sums, and exit.

4. Testing and usage

A series of technical runs to test performance, flexibility and stability of the Gatherer were carried out in years 2006-2009. The Gatherer was also tested in a series of commissioning runs. The tests were successful and showed that the Gatherer split in two tiers can handle up to 100,000 histograms coming from 10 nodes. Another test shows that the time delay between the publication of a final histogram with respect to the gathered result is less than 1 s. The Gatherer is being continuously and extensively used for the cosmics data taking and in single beam since August 2008. An example of the summed histogram of jet ϕ and η distribution is shown on Fig. 6.

5. Conclusion

An application termed the 'Gatherer' has been developed to combine monitoring results across large farms of monitoring nodes. We have implemented flexible naming, tiering and configuration

capabilities allowing this software to operate in a variety of usages in the trigger system. The Gatherer is optimized to most efficiently move results through the system and in a controlled or synchronized way, and incorporates a diverse set of operations to combine results of a variety of different types. This utility has been used extensively in online commissioning tests, including the Fall 2008 single beam running of the LHC, and meets required performance metrics for collisions running.

References

[1] T. Bold "Monitoring Using Histograms" *HLT commisioning workshop* (2006)

[2] S. Kolos "Information Service Documentation" http://atlas-tdaq-monitoring.web.cern.ch/atlas-tdaq-monitoring/IS/doc/userguide/html/is-usersguide.html

[3] S. Kolos "Online Histogramming Documentation" http://atlas-onlsw.web.cern.ch/Atlas-onlsw/oh/oh.htm

[4] Friedl, Jeffrey "Mastering Regular Expressions" *O'Reilly* ISBN 0-596-00289-0 (2002)