Workload Management

This guide describes the workload management system Vantiq uses to control each Organization's resource usage with the goal of minimizing the impact of excess resource usage by one application system on all other application systems. The workload management techniques used by Vantiq include:

- Quotas
- Rate Limiting
- Credit-based Work Management
- Buffering
- Isolated Organizations

Workload management is applied on a per Organization basis. If you have more than one application system and wish to allocate resources specifically to each application system, the application systems must be placed in separate Organizations.

Background

All application systems must contend with the problem of resource exhaustion. In most application systems, resources are implicitly managed by the physical or virtual machines on which the system is hosted. If the application system runs out of memory, it fails; if it runs out of compute resources, it slows down and, eventually, fails; if it runs out of persistent storage space, it fails.

Vantiq has the ability to run a suite of application systems on a single installation for more efficient sharing of the physical or virtual compute resources on which the Vantiq installation is hosted. However, with multiple application systems sharing compute resources, if one application system exhausts some resource, the other application systems are likely to fail because the first application system has consumed all available resources.

Vantiq implements a unique workload management capability that provides developers and operators with control over the resources allocated to each application system. This makes it possible for multiple application systems to share resources without resource exhaustion by one system impacting the availability of other systems sharing the same resources. It also makes it possible for the application system consuming excessive resources to continue operating, albeit in a degraded fashion and with possibly longer response times.

Several important benefits accrue to Vantiq's workload management capability:

- *Fair resource allocation across multiple Organizations.* For example, an organization may be provisioned with credit to utilize 20% of the available compute resources of their Vantiq installation. This leaves 80% of the compute resources available for use by other Organizations.
- *Limit the ability of any Organization to exhaust a limited resource causing the Vantiq installation to fail.* For example, an application system may activate 1,000 event handlers/second. However, the event handlers are complicated enough that the installation only has the capacity to process 500 event handler activations/second. If the application system continues to activate 1,000 event handlers/second, in the second second of operation, the system will have 1,500 event handlers active. In the third second of operation, the system will have 2,000 event handlers active. If this behavior continues, the number of active event handlers will continue to grow and the total allocated memory will continue to grow to maintain the execution state for the active event handlers. Without limits, eventually available memory will be exhausted, causing the installation to fail.
- *Produce notifications documenting excessive resource usage within an organization, simplifying root cause analysis and corrective actions.* Without workload management, excessive resource consumption may cause the installation to fail. It is difficult to diagnose an application system problem once the installation has failed. Much of the relevant diagnostic information needed to diagnose the root cause of the failure and track the problem to a particular component within a specific application system is lost at the time the installation fails. The problem is even more complex if an IT operations department controls the installation and the developers have to involve the operations team in the diagnostic effort. For developers without any access to the platform installation, the only option is to ask the platform administrator for help which may not be available in a timely fashion. With workload management, the installation does not fail. Instead, diagnostic notifications are produced explaining what application system is causing the fault and the resource that is being excessively consumed. The diagnostic notifications can be inspected within the Vantiq IDE greatly simplifying the diagnostic process.

Terminology

This description of workload management uses the following terminology:

- **application system** - A system constructed on the Vantiq platform consisting of Services, Clients, Sources, Apps, Rules, Procedures and their supporting resources.
- **Vantiq installation** - a Vantiq public cloud, private cloud or Vantiq Edge instance.
- **event handler** - a component of a Service that processes a stream of events. Generally, event handlers are constructed as either Visual Event Handlers (preferred) or VAIL event handlers. Standalone apps and rules that are not part of any Service may be created but their use is not recommended. Standalone apps and rules will be deprecated in a future release.
- **procedure** - the implementation of an operation on a Service. Standalone procedures may be defined independently of any Service. The use of standalone procedures is not recommended. Standalone procedures will be deprecated in a future release.
- **activation** - the triggering of an instance of an event handler to process an event. Activations consume compute resources and, therefore, are subject to quotas.
- **execution** - the process of executing an activated event handler.
- **quota** - a specific limit placed on either the rate at which a resource can be consumed or the maximum amount of a resource that can be consumed defined as a percentage of the total amount of the resource that is available to the Vantiq installation.

Workload Management Conceptual Model

A Vantiq installation provides compute resources to each provisioned Organization. The resources include:

- compute resources
- memory resources
- persistent storage bandwidth and capacity

For best performance, the aggregate activity across all Organizations must consume fewer resources than are allocated to the Vantiq installation. Vantiq implements a workload management strategy to fairly allocate a share of the available compute resources to each Organization.

One or more application systems are deployed within an Organization. The application systems within an Organization share the compute resources allocated to the Organization.

Vantiq workload management establishes resource consumption limits on each Organization provisioned within the Vantiq installation. The consumption limits are expressed as **quotas** that control resource consumption by managing two execution properties:

- assign maximum processing **rates** to each Organization.
- allocate a percentage of available processing resources to each organization. The allocation is expressed as the number of **credits** allocated to the organization from the pool of credits available to the Vantiq installation.
- in specific cases, limit on the amount of memory that can be allocated to in-memory state.

Maximum processing rates are applied to three resource consumption scenarios:

- the maximum rate at which messages may be read from sources
- the maximum rate at which event handlers can be activated
- the maximum length of compute intensive activities

Credit controls the total amount of work the Organization is executing at any given time independent of the rate at which new work is presented to Vantiq. In general, rates limit the amount of new work an organization can submit each second, while credit controls the resources available to execute the work submitted by the organization.

Quotas are established at the time an Organization is provisioned. Quotas can be modified at any time by the system administrator.

Observe that quotas are established on a per Organization basis. Quotas are not applied to individual application systems installed in an Organization. If the quotas for multiple application systems need to be controlled individually, each application system must be deployed in a separate Organization.

In addition to quotas, Vantiq attempts to smooth workloads that may briefly exceed the capacity assigned to them as credits. For example, an application system may be assigned a credit of 20% of the available compute resources. However, for a short period of time the workload presented to the system requires 50% of the available compute resources. Under such conditions, Vantiq will buffer the excess workload and dispatch it as active work items complete and credit again becomes available. Of course, this does not solve every problem. If an excess workload continues to be presented to Vantiq, buffer space will eventually be exhausted and the excess workload will be dropped. The application system will continue to operate, although its semantics may have been compromised by the dropped workload items.

Quotas are enforced on a per *cluster member* basis. This implies an execution rate quota of 1000/second on a three member cluster will support 1000 activations/second on EACH cluster member for an aggregate activation rate of 3000/second, assuming the activations are evenly distributed across all cluster members. Similarly, a five member cluster operating with an execution rate quota of 1000 activations/second will support an aggregate execution rate of 5000/second. These semantics also imply that adding an additional cluster member will increase the capacity available to each Organization while removing a cluster member will decrease the capacity available to each Organization.

Cluster member failure has no impact on the established quotas. However, it does reduce the capacity available to the Organization by a factor of 1/N where N is the configured number of cluster members. If the cluster is configured for three members and one fails, the cluster's processing power is reduced by 33%. The work that would normally be scheduled on the failed cluster member is reallocated to the remaining two members. If the remaining members are executing work near their quota limits, the additional work may cause the Organization to exceed its quota with the consequences described in this document. There are two mitigation strategies available for this scenario. If high reliability is essential to the application system, Vantiq recommends implementing both strategies:

- Configure the quotas with enough headroom so the workload from the failed cluster member can be taken up by the remaining members without exceeding quota limits.
- Configure the cluster with more members so that the impact of a single cluster member failing is reduced. For example, the failure of a single member in a 5 member cluster results in a 20% reduction in processing capacity while a single failure in a 7 member cluster results in a 14% reduction in processing capacity.

Persistent storage is managed in a limited manner. The application system is responsible for managing the amount of storage consumed by application types. Vantiq manages the storage consumed by the application logs by limiting the rate at which similar errors are written to the error and audit logs.

Applicability

Quotas are applied to all Organizations provisioned in a Vantiq installation in any of its deployment configurations:

- Vantiq Public Cloud
- Private Cloud, Vantiq Managed
- Private Cloud, Customer Managed
- Edge installations

However, there is one Organizational exception. Quotas and Credits **do not** apply to the *system* organization. Therefore, Vantiq **strongly** recommends application systems are **never** deployed in system administration owned namespaces on Private Clouds or Edge Installations. Vantiq Public Clouds never allow customer application systems to be deployed into the system administration organization.

Default Quotas

Default quotas are established for every Organization. The defaults can be modified by the system administrator on a per Organization basis. It is the responsibility of the system administrator to establish quotas that fairly allocate the Vantiq installation's available compute resources. If larger quotas are allocated to a number of Organizations and, in aggregate, the allocated quotas exceed the capacity of the installation, the installation may fail if the Organizations utilize their full quotas simultaneously.

A JSON document that contains the default quotas is presented below:

```
{
  "rates": {
    "execution": 1000,
    "stream" : 250000,
    "receiveMessage": 1000
  },
  "credit": {
    "default": {
        "percentage" : 20,
        "queueRatio": 2
    }
  },
  "limits": {
    "stackDepth": 200,
    "errorBreaker": {
        "sample": 20,
        "failurePercent": 80,
        "retrySample": 2,
        "retryAfter": "1 minute"
     },
    "executionTime": "2 hours",
    "synchronousIterationSize" : 100000,
    "minimumScheduledProcedureInterval" : "1 minute",
    "documentExpansion" : 0,
    "k8sResources" : {
        "vCPU"       : "0",
        "memory"     : "0",
        "gpuNvidia"  : "0",
        "gpuAmd"     : "0"
    }
  },
  "auditFrequency": "10 minutes",
  "errorReportingFrequency": "30 minutes"
}
```

A system administrator may change the quotas assigned to an Organization by editing the custom quotas JSON document associated with each Organization's configuration. This document can be viewed in the system administrator's Organization view. The custom quotas document is initially empty when an Organization is first created.

Summarizing the JSON quota document, the **rates** property contains the quotas:

- **execution** - the maximum rate at which new work items can be activated. The default is 1,000 limiting the number of event handler activations to a maximum of 1,000/second.
- **stream** - the maximum number of operations/second a compute sequence can execute. This is explicitly designed to detect infinite loops and can be ignored for practical purposes.
- **receiveMessage** - the maximum rate at which messages can be received from sources. The default quota is 1,000 messages/second received across all sources.

The **credit** property expresses the overall compute resource quota as two properties:

- **percentage** - the fraction of available compute resources that can be used by the Organization.
- **queueRatio** - the maximum number of activations that can be queued awaiting the allocation of compute resources.

The **limits** property contains the quotas:

- **stackDepth** - the maximum depth of the Vantiq execution stack. The default value is 200 which should be adequate for a typical Vantiq invocation.
- **errorBreaker** - establishes the minimum error rate at which Vantiq will suspend execution of a rule, event handler or procedure.
- **executionTime** - the maximum execution time for any event handler or procedure invocation. The default is 2 hours.
- **synchronousIterationSize** - maximum allowable size for a collection used in a synchronous FOR loop
- **minimumScheduledProcedureInterval** - lower bound for the interval of a scheduled procedure. This is an interval string (e.g.: "10 minutes") and defaults to "1 minute".
- **k8sResources** - limits on usage of K8s Resources belonging to the Vantiq cluster (*aka* `self`). If this limit is not provided, no usage is permitted. This limit is not provided by default. See Administrators Reference Guide for more details.
- **documentExpansion** – limits on the amount of memory that can be consumed for processing large objects (often, but not limited to, images) in the server. This is used to limit an organization's storage of temporary images (the results from the VIDEO source) as well as that used for sending large amounts of data (see Document Operations). The default is 0.

Finally, there are two overall quotas established to manage logging and auditing activities in an effort to not exhaust persistent storage by recording a flood of diagnostic messages:

- **auditFrequency** - the frequency with which audit entries can be produced for a given audit condition.
- **errorReportingFrequency** - the frequency with which error entries can be produced for the same error condition.

The various workload management strategies are described in more detail in the remainder of this document with chapters focused the quotas enumerated above.

Execution Rate Quota

The most common resource allocation problem encountered is application systems that activate more work than can be executed with the available compute resources. In general, Vantiq dispatches work asynchronously making it fairly easy to quickly dispatch a massive amount of work. For example, if an application system is receiving 100,000 messages/second and dispatching an event handler to process each message, 100,000 event handler activations occur each second. In a large installation this will not be a problem but, in a smaller installation, this workload may overwhelm the Vantiq installation. Vantiq will do everything in its power to successfully execute the workload by first scheduling as much work as possible. If all the work cannot be scheduled, the excess work will be buffered pending the availability of resources. However, if the queues overflow, the Vantiq installation has no choice but to drop some of the pending load to keep the application system from failing.

The **execution** quota establishes the maximum rate at which an Organization can activate event handlers. The execution quota helps solve the problem of creating activations faster than they can be completed. The default quota is 1,000 activations/second. Quotas are established and enforced on a per *cluster member* basis. If the Vantiq installation contains three cluster members and a quota of 1,000 activations/second, the aggregate quota is 3,000 activations/second assuming the activations can be distributed evenly across the three cluster members. The activation rate is computed using the sum of the requested activations across all event handler activations on a cluster member. There is not a separate quota for each event handler.

If the Organization as a whole attempts to activate event handlers at a higher rate than the aggregate quota, the excess activations are buffered. When the execution rate quota is exceeded, an audit record is written to the error log documenting the fact the system is now buffering event handler activations. An example of such an audit record when an execution rate quota of 250 is exceeded:

> Audit Alert: The execution rate quota of 250.0/s for organization butterworth has been exceeded. As a result the events for rule are being buffered.

This quota was exceeded on a cluster with three members implying the aggregate rate at which event handlers were being activated was greater than 750/second. In this example, the activation rate was roughly 1,500/second at the time this audit record was recorded.

You can also monitor buffering activity using the Grafana Service Execution Dashboard, App Execution Dashboard, or Rule Execution Dashboard, depending on the types of resources defined in the application system being monitored.

It may be the case the default execution rate quota of 1000 invocations/second is inadequate to support a specific application system or set of application systems. In such cases, the execution quota can be increased if there are sufficient compute resources available to execute event handlers at a higher rate. However, Vantiq recommends that the rate be limited to a maximum of 20,000 activations/second in moderate size Vantiq installations. In larger installations, the maximum rate is only limited by the resources available to process the event handler activations.

Application systems under development typically exhibit excessive event handler activation rates when one event handler invokes more than one additonal event handler. For example, below is a sketch of an event handler that selects a set of democontrol objects. For each democontrol object selected, a set of sampledata objects are selected. For each sampledata object selected, another event handler is activated. If democontrol returns 5 objects and sampledata returns 5 objects, each activation of the explodingWork event handler will produce 25 additional activations of *explodingWork*. Needless to say, after a few event handler activations, the execution rate limit is exceeded. The script for the explodingWork event handler:

```
RULE explodingWork
WHEN PUBLISH OCCURS ON "/doWork"

SELECT * FROM democontrol {
    SELECT * FROM sampledata {
        PUBLISH {} TO TOPIC "/doWork"
    }
}
```

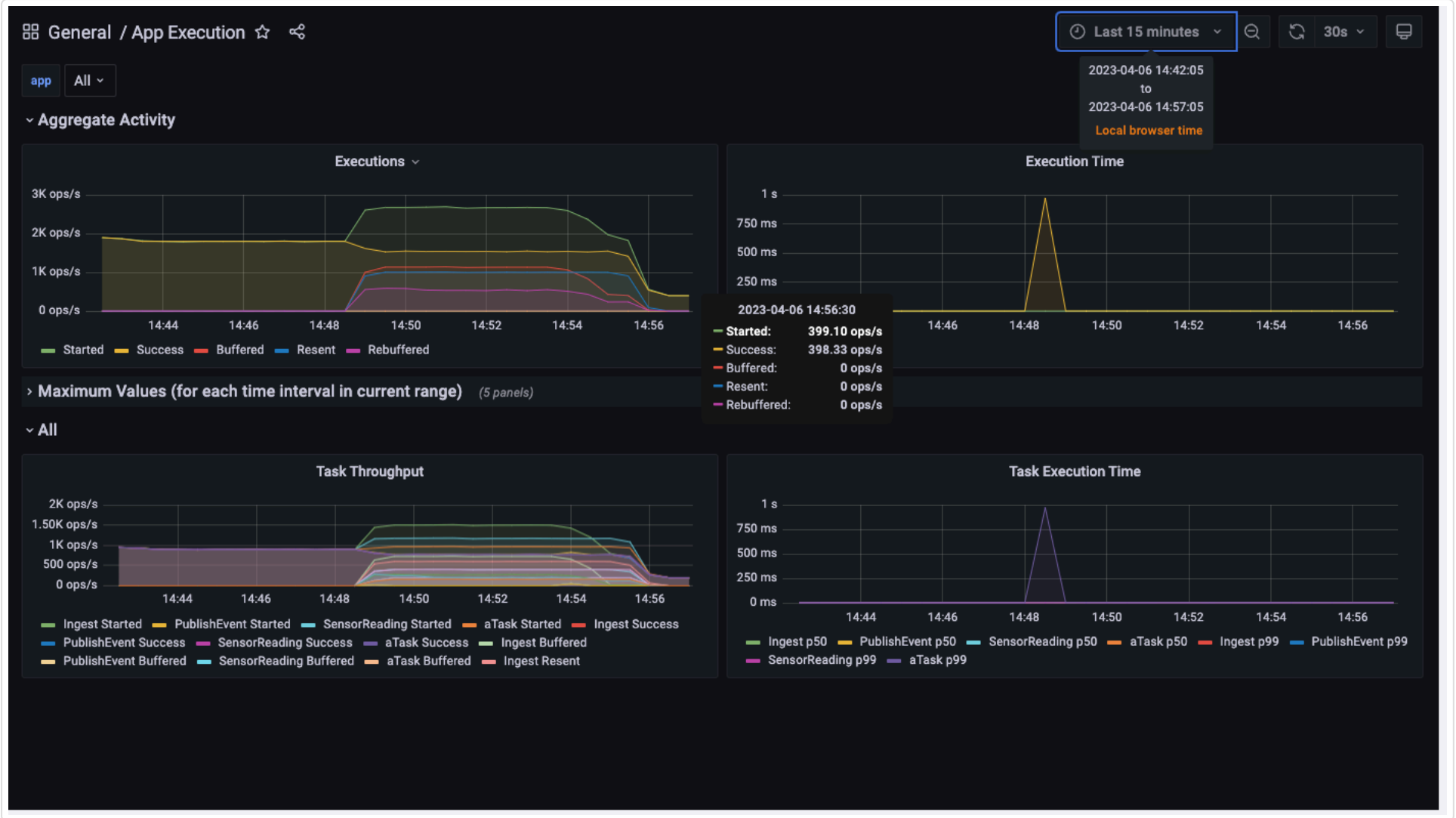
Another more limited example can be found by creating a design model from the IoT template. By default, the template produces an application system with an event handler that ingests a stream of events from an external source–the ingested events are then published to an event handler that processes each ingested event. By default, this handler publishes two additional events to process the events through a Transformation followed by a ComputeStatistics task. Thus, each inbound event produces four event handler executions. If events arrive at 1000 events/sec, the aggregate execution rate quota required to process the workload is roughly 4000/3 = 1,334. Some headroom will be required so a minimum recommended quota for this workload might be 1,500/second.

Diagnostics

Execution rate quota issues are most easily diagnosed using the Grafana dashboards. The principal dashboards showing execution rate quota violations is the Service Execution or App Execution dashboard, which display a summary of task execution statistics. Below is an example of the App Execution dashboard showing a system that was running at roughly 1900 execution/sec from 14:44 to 14:48 without any quota violations. At 14:48 we artifically reduced the execution rate quota from 1,000 executions/sec to 250 executions/sec. Since the system has three servers, it could support roughly 3,000 executions/sec while the system was executing 1,900 executions/sec. With the quota reduced to 250 executions/sec, the system is limited to roughly 750 executions/sec - 250 executions/sec on each server. At 14:49 the impact of the reduction is seen in the additional data displayed indicating the system is now buffering roughly 1,150 executions/second, resending roughly 1,000 executions/second and rebuffering roughly 600 executions/second. These numbers can be interpreted as follows:

- 1,150 buffered executions - the system is limited to roughly 750 executions/second and we are starting 1,900/second. The excess executions, 1,150 of them, are buffered waiting for quota availability.
- 1,000 resends - the system attempts to resubmit buffered executions on a periodic basis. This indicates they are being resumbmitted at roughly 1,000 buffered requests/second.
- 600 rebuffered - not all the resubmitted requests can be honored and stay within the execution rate quota. When the resubmission is declined, the message is re-inserted into the buffer and will be resumbitted at a later time.

The corresponding Grafana dashboard is displayed below:



Mitigations

If an Organization attempts to activate work at a rate faster than the execution rate quota, Vantiq will buffer the excess activations. The buffered activations are scheduled for execution when compute resources become available on a first in, first out basis.

Buffer activity can be observed in the Grafana Application Execution Dashboard and the Rule Execution Dashboard depending on the type of the resources defined in the application system.

In general, well-designed application systems should experience little or no event handler activation buffering. If an event handler activation is buffered, the activation of that event handler is delayed until compute resources are available. The delay may cause the application system to behave improperly. If such delays cannot be tolerated, it is critical the application system be redesigned to eliminate any buffering when deployed in its production configuration.

Buffering associated with the execution rate quota occurs because too many activations are being requested each second. This may occur because:

- the application system is improperly constructed and is producing excessive activations (see the explodingWork example above).
- the application system experiences spikes in the arrival of events causing it to periodically exceed the execution quota.
- the application system is processing a large number of events and each event activates additional event handlers.

The first problem is typically mitigated by thoroughly inspecting the application system's implementation looking for implementation errors or poor design choices. Specifically, a design that creates exponentially more activations with each event handler execution is likely to be incorrectly designed.

The second problem is addressed by making sure the system can tolerate delays in activating rules when the system is presented with too large a workload resulting in the excess workload being buffered. This will implictly smooth the rate at which event handler activations are produced. Highly scalable application systems are most efficient if they are presented with a predictable workload. Spikes are difficult to handle because the application system and the underlying Vantiq installation must be provisioned to handle the largest possible spike. In some cases, the spike may be two or three orders of magnitude larger than the average workload. Such provisioning is not cost effective. The optimal strategy is to accept the work associated with the spike but spread the execution out over a longer time frame. For example, if a spike occurs once every 5 minutes, it may be possible to accept the work associated with the spike but then process it over the ensuing 5 minute interval before the next spike occurs. This is exactly what buffering accomplishes.

The third problem is typically mitigated using one of the following techniques:

- Raise the execution rate quota. This applies if there are additional execution resources available or if the Vantiq installation can be provisioned with additional resources.

- Revise the application system design to reduce the number of activations. For example, the rate at which events are ingested might be reduced by packing several events into a single composite event. Such adaptations of the application system design can be quite effective but, for a given application system, may not be practicable.

In some advanced cases, the default buffering semantics are suboptimal and a more optimal solution can be realized by taking advantage of the semantics of the workload to provide more sophisticated buffer management. For example, it might make sense to buffer some work on a high priority queue and some work on a low priority queue. In order to do this, the application system should be designed to manage the queues itself and implement a dispatcher to perform activations in conformance with the priorities established by the application system semantics.

Execution Credit Quota

The execution credit quota complements the execution rate quota by establishing a maximum number of executions an Organization can have outstanding at any moment in time. Active executions consist of executing event handlers. Credit limits are specified as a percentage of the total available resources that can be consumed by the Organization. For example, if the Vantiq installation overall supports 1,000 execution credits and an Organization has a credit limit of 25%, the Organization can use up to 250 execution credits at any moment in time.

Execution credit quota is applied to Vantiq cluster members. Thus, if the execution credit quota is 25%, the Organization can consume 25% of each cluster member. The number of execution credits supported by a Vantiq installation depends on the compute capacity of the underlying cluster members. As a general rule, each cluster member (VM) supports roughly 400 credits per core. That implies a four core VM has 1600 credits available while an eight core VM has 3200 credits available.

Credits are allocated across all Organizations hosted on the Vantiq installation. It is never recommended to give every Organization a credit limit of 100% as this is the equivalent of not establishing any credit limit. Each Organization will be able to use the full capacity of the machine.

Credit limits try to allocate resources fairly. For example, if two Organizations are assigned credit limits of 100% and both activate enough work to utilize their full credit allotment, each Organization will be restricted to utilizing 50% of the available credit. In such scenarios, the credit limit is of limited utility because it cannot guarantee the availablility of the full credit allotment to any Organization.

However, if the two Organizations operate their applications at different times of the day and their activities do not overlap, each could be given a credit limit of 100% since there will be no actual conflicting resource demands from the two Organizations.

Setting credit limits is a challenge for the system administrator, who must know the expected behavior of each Organization in order to give them an adequate credit limit while still allocating credit fairly among all provisioned Organizations. The system administrator periodically reviews the various metrics available in the Vantiq IDE's Grafana dashboards to gain a more accurate understanding of the resources being consumed by each Organization.

If an Organization's credit limit is exhausted, additional activations will be buffered pending credit availability. When credit is available, buffered entries will be activated. The buffer is activated using first in, first out semantics.

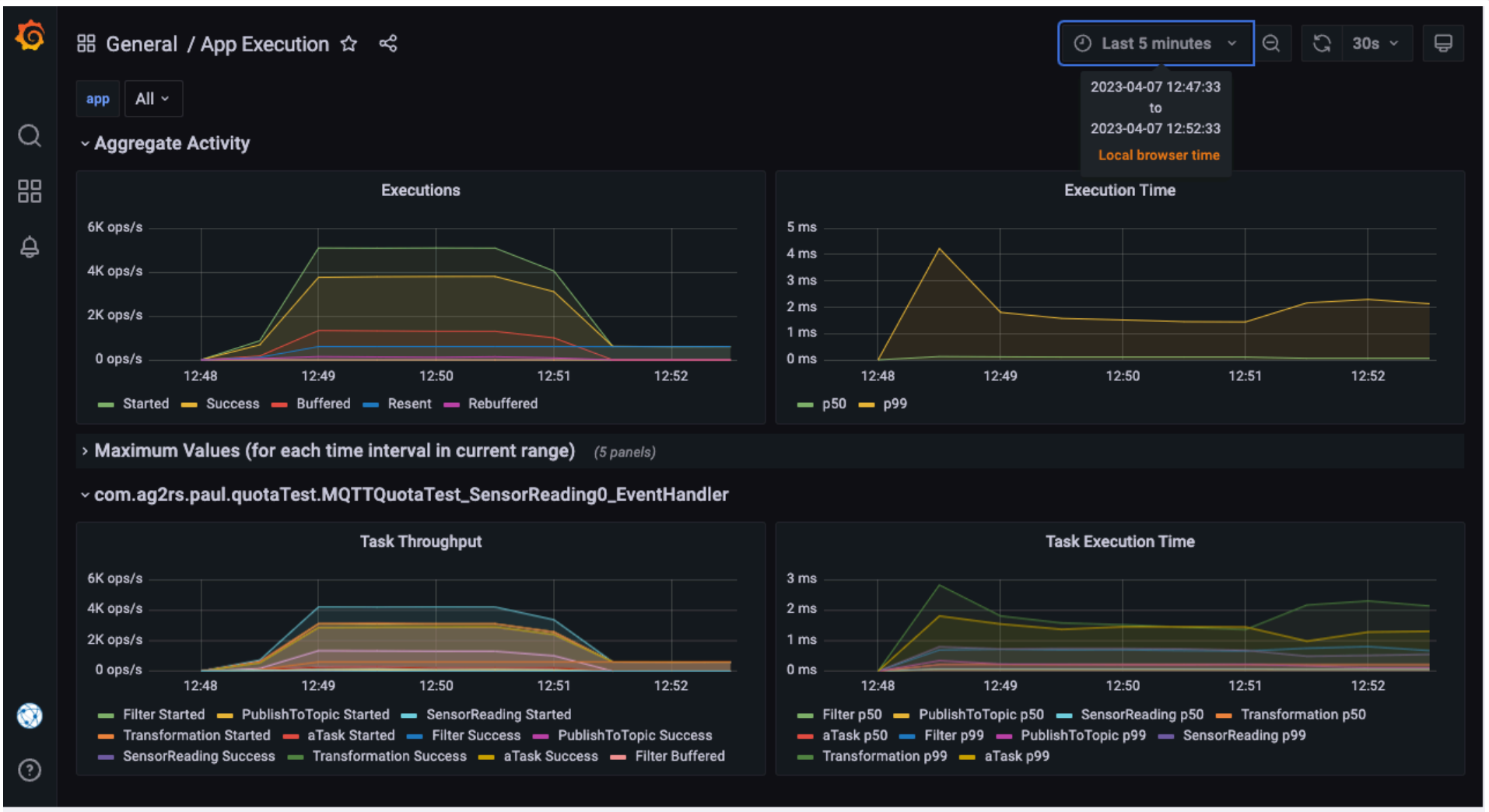
Exhausting the credit limit causes the following action by the Vantiq installation:

- if the requested work was submitted as a REST request, the request is rejected returning a status of 429.
- if the requested work was submitted as a Websocket request, reading from the socket is stopped, backpressuring the network and the client. This may cause other asynchronous requests sent via the same socket to be delayed.
- if the request originated in VAIL script, the request is buffered delaying the execution of the VAIL script and increasing its latency. If the buffer space is exhausted, the request will dropped.
- if the request is an event handler activation, the activation is buffered.

A audit record is written to the error log when execution credits have been exhausted and Vantiq starts to buffer subsequent execution requests. The Grafana Application Execution and Rule Execution dashboards can be used to monitor buffering activity.

Diagnostics

Exceeding the execution credit quota is most easily diagnosed using the Grafana App Execution dashboard which displays task start rates and completion rates as well as any buffering activity caused by exceeding either the execution rate quota or the execution credit quota. The following App Execution dashboard presents a system that is exceeding its execution credit limit.



The system was started at 12:48 and quickly ramped up to an execution rate of roughly 5,100 executions/second. However, the execution credit only allowed 3,800 simultaneous executions. This caused Vantiq to immediately start buffering 1,300 executions/second. The event stream driving the system was stopped at about 12:50:30 causing the execution rate to diminish until at 12:51:30 the inbound event stream was stopped. The system then continued to start previously buffered executions. Buffered executions are continuing to be started at 12:52:30 which is then of the time interval displayed in the Grafana dashboard. Buffered executions continued to be started until 12:54 at which time the buffer was empty.

The system can tolerate relatively short bursts of buffering but, if the example system above was allowed to continue to run indefinitely, eventually subsequent executions would be dropped.

Mitigation

Exhausting the *credit limit* will cause activations to buffer. Exhausting the *buffer space* will cause activations to be dropped. Since it is assumed that all work is essential to realizing the correct behavior of the application, buffer exhaustion conditions must be investigated and corrected. In general, the most likely problem is the work consuming the credit is taking longer to execute than predicted. The Grafana dashboards in the Vantiq IDE can be used to review the execution times for Event Handlers and Procedures to find ones whose elapsed execution time is longer than expected or longer than can be tolerated by the volume of work the application system is requesting.

As a rule of thumb, for highly scalable application systems, any event handler or procedure whose p99 elapsed time is greater than 1 second is an immediate candidate for review. Elapsed times can be viewed in the Grafana dashboards. Grafana displays two elapsed times by default: the p50 time, which is the median response time, and the p99 response, which represents the 99th percentile response time. In general, scalable systems are processing hundreds or thousands of events/second and any execution that takes longer than a second is likely to cause ever increasing amounts of new work to queue waiting for credit, eventually causing a credit limit violation. A common problem is that the event handler or procedure is either accessing a database or an external system that is slow to respond, causing elapsed times to increase while it waits for a response to a request. The application system designer should review the database or external system access to:

- confirm access is necessary. The best solution is to remove the offending activity. For instance, it might make more sense to cache commonly used reference data than to access it from the database for each event processed.
- confirm the activity is running as expected. For example, queries may run slowly if the underlying dataset is not properly indexed. Many external systems are not designed to be scalable and exhibit high latencies under even moderate loads.
- consider whether the offending activity might be better executed asynchronously. Database updates might be converted into events that are processed by a separate database update event handler that can execute asynchronously. Such an event handler has the opportunity to optimize update processing. However, note that creating an asynchronous execution activity consumes credit when it is activated.

Additional Buffer Semantics

Event handler activations that exceed the execution rate quota or the execution credit quota are buffered as described in the previous two sections. This section describes additional buffer semantics an application system designer should consider as part of the application system design process.

Event handler execution buffers are first in, first out (FIFO) queues with individual buffers allocated to each event handler for which execution credit is unavailable. Separate buffers for each event handler are maintained on each cluster member. When activations are currently buffered and credit becomes available on a cluster member, the oldest entry in the current cluster member's queue is activated. However, the following condition may arise causing the entry to be re-inserted at the end of FIFO buffer:

- Vantiq removes the entry from the buffer.

- Vantiq attempts to activate the entry but, before the entry can be activated, all available credit is again consumed making it impossible to perform the activation.
- Vantiq re-inserts the entry at the end of the FIFO buffer.

The event handler activation buffers on each cluster member store up to 100 MB of event handler activations for each event handler. However, if the buffer space is exhausted, new activations cannot be buffered and are dropped. Drops are most likely to occur if the activation rate is sustained at a level above the event activation quota or above the execution credit quota described in the previous two sections. Care must be taken to assure buffering only occurs due to transient increases in event handler activations and not because the system has been designed to regularly process event handler activations at a rate higher than either quota.

The 100 MB buffer size for each event handler is an installation-wide configuration parameter and may only be changed at Private Cloud or Edge installation time. If an installation elects to increase the buffer size, care must be taken to allocate enough storage space to the Vantiq database to accommodate all the event handler buffer space that will be used by all the event handlers across all provisioned Organizations executing within the installation.

Dropped events can be monitored in the Grafana Event Processing dashboard.

If a cluster member fails with entries in its event handler activation buffer, the entries are deleted (dropped) when the cluster member is restored.

receiveMessage Quota

The **receiveMessage** quota limits the number of messages sources can receive. It is specified as a rate and the default value is 1,000 messages/second. If messages arrive at a rate higher than the receiveMessage quota, *the offending messages are dropped*. If the application systems owned by an Organization must receive messages at a higher rate than the receiveMessage quota, the receiveMessage quota must be increased.

It is common for the receiveMessage quota to be increased for larger application systems. Message rates of thousands to tens of thousands are seen in larger systems. Receiving messages is fairly low overhead but each message received generally activates additional processing that may be expensive. Therefore, before raising the receiveMessage quota, it is important to determine whether the Vantiq installation has enough processing capacity to support the expected load. In addition, the execution quota typically must also be raised to support greater than 1,000 event handler invocations/second.

Messages are generally received from Sources. A Source, such as an MQTT, Kafka or AMQP source, subscribed to one or more topics is assigned to a cluster member and will only receive messages on that cluster member. This means the receiveMessage quota generally applies to a single cluster member. Thus, if the receiveMessage quota is 1000, a single Source will be limited to receiving 1000 message/second since they are all received on a single cluster member. If there are other Sources assigned to other cluster members, the total messages/sec for the cluster can be more than 1000 messages/sec. All the Sources assigned to a single cluster member, however, will still only be able to receive an aggregate of 1000 messages/sec.

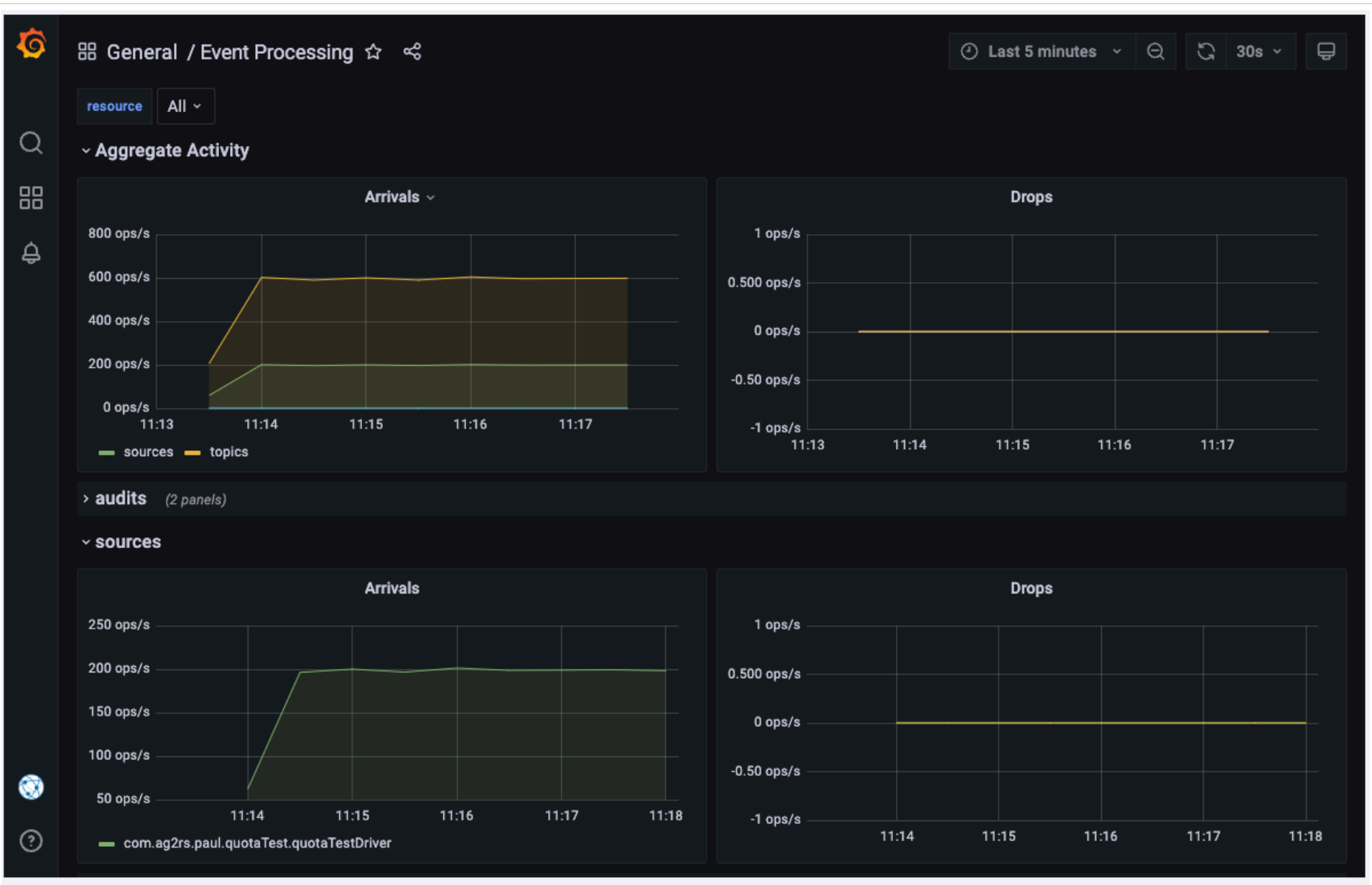
Diagnostics

receiveMessage quota issues are diagnosed using two mechanisms:

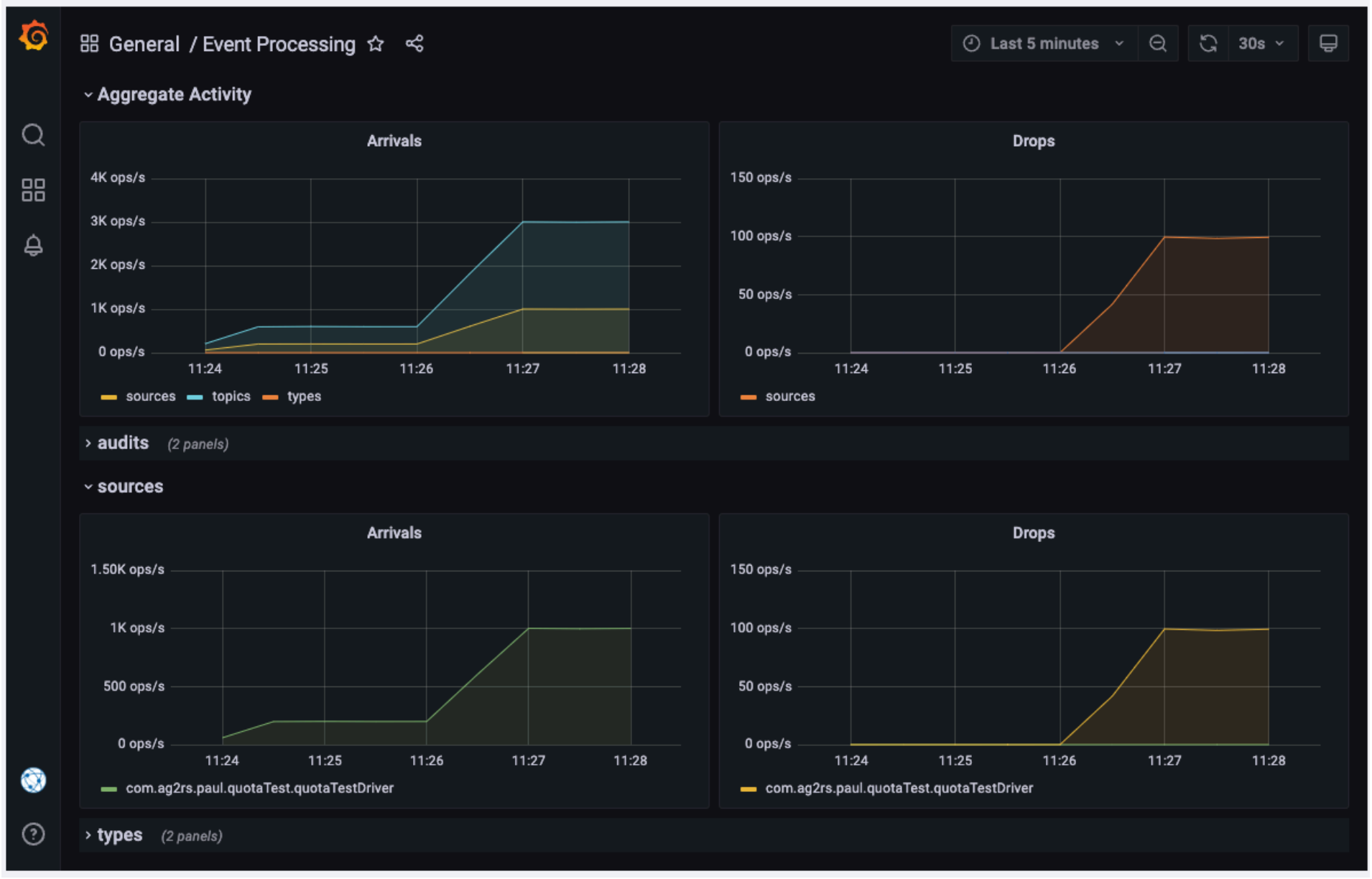
- Grafana dashboards
- Error logs

The most relevant Grafana dashboard for detecting receiveMessage quota violations is the **Event Processing** dashboard. The **Aggregate Activity** section displays the rate at which messages are arriving from all sources in the left panel and any drops that have occurred becaue the receiveMessage quota has been exceeded in the right panel.

Here is a snapshot of the Event Processing dashboard when messages are arriving at approximately 200 messages/sec, well within the 1,000 message/sec quota assigned to this Organization.



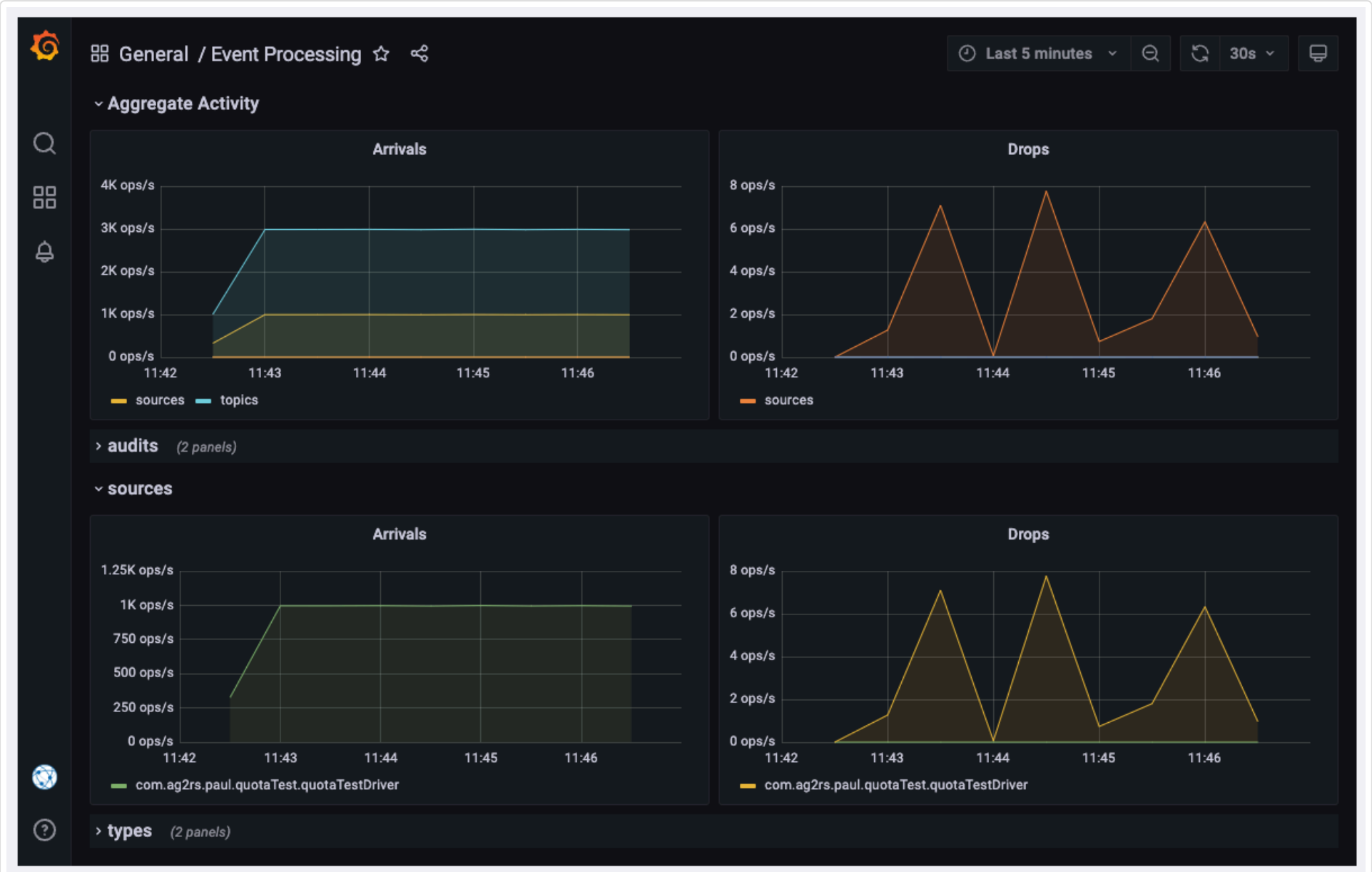
Here is a snapshot of the event processing dashboard when the receiveMessage quota is being exceeded because messages are being delivered by a source at roughly 1,100 messages/second while the quota remains at 1,000 messages/second.



Observe the left hand panel is showing an arrival rate of approximately 1,000 messages/sec which is the receiveMessage quota. The right hand panel is showing messages being dropped at a rate of roughly 100 messages/second which is the number of messages that exceed the receiveMessage quota.

The same dynamic is illustrated in both the Aggregate Activity graph at the top of the display and the lower display that shows the detailed rate at which each source is receiving messages. Since this example uses a single source, the upper and lower graphs show identical information.

A final issue to consider is that message arrival rates may vary slightly due to the vagaries of networking and process scheduling. Therefore, if the receiveMessage quota is set to 1,000 messages/sec and messages are being delivered at an average rate of 1,000 messages/second, there is a high probability that a few messages will be dropped from time to time as the instaneous arrival rate exceeds the average. This is illustrated in the following Grafana dashboard depicting a situation in which the average arrival rate is 1,000 messages/sec:



The right hand display of drops shows arriving messages are being dropped at a rate between 0.5 messages/sec and roughly 8 messages/sec illustrating that actual message arrival rates should be designed to be less than the receiveMessage quota. For example, if we move the actual message arrival rate to an average of 950 messages/sec, the drops are eliminated.

receiveMessage quota violations are also logged in the error log. Here is an example of such a log entry:

Execution Start Time: 2023-04-05 11:57:01.644 | **Elapsed Time:** 0ms

Message: Audit Alert: The message rate quota of 1000.0/s for organization butterworth has been exceeded. As a result the source com.ag2rs.paul.quotaTest.quotaTestDriver is being throttled.

Such errors indicate the system is exceeded its quota and messages are being dropped. Action must be taken to correct the problem as described in the next section on mitigations.

Mitigation

The most common strategies to mitigate receiveMessage quota violations:

- increase the receiveMessage quota if the semantics of the application system require a higher message rate.
- pack multiple events into a single message if the application semantics support such behavior. However, in some sense this strategy increases the likelihood of execution rate quota violations as the inbound message generally invokes an event handler to process each event packed into the message.

Quota Interactions

The execution rate quota, the execution credit quota and the receiveMessage quota work together as the primary workload managers. This section provides some insight into how these quotas work together.

The receive message quota limits the rate at which messages can be received from sources by dropping messages that exceed the receiveMessage quota. The arrival of each message activates one or more event handlers. These event handler activations are subject to both:

- the execution rate quota that limits the maximum activation rate
- the execution credit quota that limits the amount of work that can be active at any point in time.

If either the execution rate quota or the credit quota are exceeded, the activations are buffered.

These semantics make it crucial that the application system designer keep these quotas and their interactions in mind. Effectively, the application system should be designed so that it does not exceed the receiveMessage quota as this condition will cause work to be dropped. The application system can exceed the execution rate quota and/or the execution credit quota with the work only deferred until credit is available. As long as the credit quota is not exceeded by too large an amount the application system should run as expected. However, if the number of event handler activations is very high, the execution buffers will overflow causing work to be dropped and the application to behave incorrectly. Care must be taken to ensure the application does not exhaust the available buffer space.

stackDepth Quota

The stackDepth quota specifies the maximum depth of the VAIL execution stack. The limit is primarily designed to detect coding errors like *infinite recursion*, as deeply nested execution is rare in a well-designed Vantiq application system. The stack depth is increased by one each time an event handler or procedure invokes a procedure. It is unlikely the stackDepth limit will be exceeded unless the application contains a potentially deeply nested recursive procedure. Given the classes of data usually processed by real-time, event-driven systems, it is unlikely deep recursion will occur in a typical application system.

Vantiq recommends leaving stackDepth at its default value. It is very rare for Vantiq procedures to include recursion that places more than 200 procedure contexts on the stack.

executionTime Quota

The executionTime quota specifies the maximum elapsed time for an event handler or procedure execution. If the execution runs for longer than the executionTime limit, the execution is terminated and an error is reported in the error log. It is not uncommon for GenAI algorithms to run for a substantial amount of time, either due to the sheer processing involved or because they are waiting for input from a intermittent source (such as the user). For this reason, the default execution time limit is configured to be 2 hours.

> The real-time, event driven portions of an application should still be designed to respond quickly and will typically complete long before this limit is applied.

Each activation of an event handler executes asynchronously. An activation may cause cause another activation at a later time. Since the second activation is executed asynchronously with respect to the first activation, the executionTime limit does not apply to the elapsed time required to run both activations.

errorBreaker Quota

The errorBreaker quota is designed to stop the execution of event handlers, rules and procedures that fail continuously. The assumption is that they cannot be operating correctly if they generate a large number of failures. If they are operating correctly and generating a large number of failures, they should be restructured so that they do not generate failures.

Common cases under which an event handler, rule or procedure fails continuously include:

- The event handler, rule or procedure communicates with an external system and the external system has failed or communication with the external system has failed.
- The event handler, rule or procedure contains a logical error that causes it to fail under conditions that were not tested but that occur frequently.

The errorBreaker limit specifies the conditions under which failing event handlers, rules or procedures are no longer executed. There are 4 properties that describe the errorBreaker limit:

- **sample** - the number of invocations of the event handler or procedure that must have occurred before the errorBreaker is evaluated. This eliminates the premature activation of the errorBreaker. The default value is 20 executions.
- **failures** - the percentage of executions that result in a failure. The default value is 80% of the executions. If there have been 20 executions and at least sixteen of them have resulted in failures, the errorBreaker will be triggered and subsequent activations or invocations will no longer be scheduled for execution.
- **retrySample** - once the errorBreaker has been triggered, how many activations or invocations must occur before re-evaluating the status of the errorBreaker. The default value is 2 samples.
- **retryAfter** - once an errorBreaker trips, how long the system should wait before re-evaluating the status of the event handler, rule or procedure. The default value is 1 minute.

Vantiq recommends leaving the default values as is.

k8sResources Quota

The k8sResources values set limits on usage of K8s Resources belonging to the Vantiq cluster (*aka* `self`). Using these settings requires the cloud to be configured with a k8sworker process running. Do not specify this setting if that is not configured. See the External Lifecycle Management Guide for more details on the K8s Worker.

If this limit is not provided, no usage is permitted. This limit is not provided by default (that is, it is `null` by default). See the Administrators Reference Guide for more details.

The format of the k8sResources settings are:

```
"k8sResources" : {
        "vCPU"       : "0",
        "memory"     : "0",
        "gpuNvidia"  : "0",
        "gpuAmd"     : "0"
}
```

Stream Quota

The default quotas include a **stream** quota that defaults to 250,000 operations/second. The stream quota limits the length of compute intensive activities and is primarily used to detect long running loops. Long running compute activities will block an execution thread, negatively impacting the scheduling of real-time event handlers. There is little reason to adjust this quota as all reasonable event handlers and procedures execute at rates well under the default quota.

Detailed Credit Quotas

The default credit limit applies equally to all kinds of credits. However, it is possible to assign unique credit values to each kind of credit. The kinds of credits are:

- *ExecutionManager* - this is the most important kind of credit controlling the amount of simultaneous compute activity an Organization can perform.
- *FederationManager* - controls the number of simultaneous federation requests. Federation requests are requests that cross Namespace and/or Organization boundaries.
- *ModelManager* - controls the number of simultaneous model updates. The model contains the metadata that decribes the Organization's application systems.
- *RuleManager* - controls the number of simultaneous Rule updates.
- *SecurityManager* - controls the number of simultaneous requests to the security manager to modify Role definitions.
- *SourceManager* - controls the number of simultaneous requests to the Source manager to modify source definitions.

Vantiq recommends that credit limits other than the ExecutionManager Credit Limit be left at the default value. They are rarely exceeded.

auditFrequency Quota

The auditFrequency defines how often an error report for the same violation will be written to the audit trail. This keeps the audit trail from filling with duplicate messages if the same audit condition is encountered frequently. For example, messages may be dropped if the invocation rate exceeds the Organization's execution quota. In scalable applications such a condition may occur hundreds or thousands of times a second if the application system is improperly designed. Thousands of identical errors in the audit trail will not help with problem diagnosis. Therefore, the system limits the audit records to one per error type during each auditFrequency interval.

Be aware this implies a single diagnostic for dropped messages or other rate or credit violations may actually represent a large number of identical failures that have occurred in a single auditFrequency interval.

Vantiq recommends operating with the default auditFrequency.

errorReportingFrequency Quota

Similar to auditFrequencey, errorReportingFrequency determines how often an error report for the same type of error violation will be recorded in the error log. This reduces the chances of overflowing the log if a specific error occurs continuously, making the overall application system far more tolerant of error storms.

However, be aware that if you see an error in the error log, the diagnostic may actually represent a large number of identical errors that have occurred in a single errorReportingFrequency interval.