Improving event distribution in Splunk Enterprise

Event distribution is how Splunk Enterprise spreads its incoming data across multiple indexers, and therefore underpins linear scaling of indexing and search. It is critical for the even distribution of search (computation) workload. Bad event distribution is when the spread of events is uneven across the indexers. It causes search time to become unbalanced, searches to take longer to complete, and a reduction in throughput. The Splunk Enterprise randomized round robin algorithm of indexer selection quickens event distribution because the first indexer isn't always performing the bulk of the effort. However, many factors can lead to problems in event distribution, and consequently, search degradation.

You can use the REST API to see how much data all your indexers are ingesting. You want to see a fairly flat bar chart and a low standard deviation.

```bash
| rest /services/server/introspection/indexer | eventstats stdev(average_KBps) avg(average_KBps)
```

You can also examine ingestion trends over time to look for outliers.

```bash
index=_internal Metrics TERM(group=thruput) TERM(name=thruput) sourcetype=splunkd
| dbinspect index=* | stats values(splunk_server) as indexer
| eval host_count=mvcount(indexer), search="host IN ("mvjoin(mvfilter(indexer!="")), ", ").""]
| eval host_pipeline=host."-".ingest_pipe
| timechart minspan=30sec limit=0 per_second(kb) by host_pipeline
```

Finally, you can count events per indexer and compare across all indexers. Before running the following search, update the index and splunk_server fields with your indexes, sites, or clusters. You can also group the stats by host instead of index. The search provides an event distribution score (normalized standard deviation) for each index. In a single-cluster site, this score should be approximately the same for each indexer. However, in a multi-site or multi-cluster environment, groups of indexers are likely to have different numbers of events.

```bash
| tstats count WHERE index=* splunk_server=* host=* earliest=-5min latest=now BY splunk_server index
| stats sum(count) AS total_events stdev(count) AS stdev avg(count) AS average BY index
| eval normalized_stdev=stdev/average
```

You can also use this event distribution dashboard to find key metrics, such as the variation of events across indexers, how many indexers received data in a specific time range, and whether event distribution is improving over time.

---

**Find sticky forwarders**

Universal forwarders rely on natural breaks in application logs to break up the data. Intermediate universal forwarders work with unparsed streams, which means that some connections can last for hours.
To solve this problem, turn on the `forceTimebasedAutoLB` setting in `outputs.conf` to force existing data streams to switch to a newly elected indexer every auto load balancing cycle. A load balancing cycling controls for how long (the `autoLBFrequency` setting) or for how much volume (the `autoLBVolume` setting) a forwarder sends data to an indexer before redirecting outputs to another indexer in the pool. The `autoLBFrequency` default is 30 seconds, which is often too long, and the `autoLBVolume` default is 0, while 1 MB is a good value to start with. These two settings work together. The forwarder first uses `autoLBVolume` to determine if it needs to switch to another indexer. If the `autoLBVolume` setting is not reached, by the time the `autoLBFrequency` limit is reached, the forwarder switches to another indexer. However, when turning on the `forceTimebasedAutoLB` setting, note that forcing a universal forwarder to switch can create broken events and generate parsing errors.

To find sticky forwarders, run the following search:

```
index=_internal sourcetype=splunkd
   TERM(eventType=connect_done) OR TERM(eventType=connect_close)
   | transaction startswith=eventType=connect_done endswith=eventType=connect_close
   sourceHost sourcePort host
   | stats stdev(duration) median(duration) avg(duration) max(duration) BY sourceHost
   | sort - max(duration)
```

**Find super giant forwarders**

When an intermediate forwarder aggregates multiple streams, it creates a super giant forwarder. One forwarder that is handling most of the effort is a data imbalance issue and should be reconfigured through at least one of the following methods:

- Configure `EVENT_BREAKER` and / or `forceTimeBasedAutoLB`.
- Configure multiple pipelines and validate that they are being used.
- Configure `autoLBVolume` and / or increase switching speed, while keeping an eye on throughput.
- Use `INGEST_EVAL` and `random()` to shard output data flows.

To find a super giant forwarder:

```
index= _internal Metrics sourcetype=splunkd
   TERM(group=tcpin_connections) earliest=-4hr latest=now
   [ | dbinspect index= _* 
      | stats values(splunk_server) AS indexer
      | eval search= "host IN ("mvjoin(mvfilter(indexer!='"'), ", " ))"
      | stats sum(kb) AS throughput BY hostname
      | sort - throughput
      | eventstats sum(throughput) AS total_throughput dc(hostname) AS all_forwarders
      | streamstats sum(throughput) AS accumulated_throughput count BY all_forwarders
      | eval coverage= accumulated_throughput/total_throughput, progress_through_forwarders= count/all_forwarders
      | bin progress_through_forwarders bins=100
      | stats max(coverage) AS coverage BY progress_through_forwarders all_forwarders
      | fields progress_through_forwarders coverage
```

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
Resolve indexer abandonment

A common cause for abandonment is forwarders only able to connect to a subset of indexers due to network problems, such as a firewall blocking the connection. This is common when the indexer cluster is increased in size.

Run this search to look for forwarder errors that indicate this problem is happening:

```
index=_internal earliest=-24hrs latest=now sourcetype=splunkd
TERM(statusee=TcpOutputProcessor) TERM(eventType=*)
| stats count count(eval(eventType="connect_try")) AS try
count(eval(eventType="connect_fail")) AS fail
count(eval(eventType="connect_done")) AS done BY destHost destIp
| eval bad_output=if(try=failed,"yes","no")
```

Add targets to a forwarder

Forwarders can only send data to targets that are in their lists. Targets are often excluded from lists when the indexer cluster is increased in size. Encourage customers to use indexer discovery on forwarders so this never happens.

Run the following search to learn which forwarders fail to connect to a target. Note that the search assumes all indexers have the same forwarders, which might not be true with multicladsters and multisites.

```
index=_internal earliest=-24hrs latest=now sourcetype=splunkd
TERM(eventType=connect_done)
TERM(group=tcpin_connections)
| dbinspect index=*  
| stats values(splunk_server) AS indexer
| eval host_count=mvcount(indexer),
search="host IN (".mvjoin(mvfilter(indexer!=""), ", "").")"
| stats count BY host sourceHost sourceIp
| stats dc(host) AS indexer_target_count values(host) AS indexers_connected_to BY sourceHost sourceIp
| eventstats max(indexer_target_count) AS total_pool
| eval missing_indexer_count=total_pool-indexer_target_count
| where missing_indexer_count != 0
```

Resolve starving forwarders

Sometimes an indexer or an indexer pipeline has periods when it is starved of data because there are not enough incoming connections. It continues to receive replicated data and search replicated cold buckets, but it does not search hot buckets.

Run this search to see if the indexers in the cluster have obviously varying numbers of incoming connections. If so, you can fix the problem by increasing the switching autoLBFrequency setting in `outputs.conf` and the number of pipelines on the forwarders.

```
index=_internal earliest=-1hrs latest=now sourcetype=splunkd
TERM(eventType=connect_done) TERM(group=tcpin_connections)
| dbinspect index=*  
```
Next steps

The content in this article comes from a [Conf19 Talk](#), one of the thousands of Splunk resources available to help users succeed.