Optimizing search in Splunk Cloud Platform

Slow searches can be caused by inefficient search practices, but they can also be caused by poor data quality. Inefficiencies such as incorrect event breaks and time stamp errors in the data can cause indexers to work overtime both when indexing data and finding the search results. You want to resolve these issues to get performance improvements.

Use the Monitoring Console

Use Splunk Cloud Platform Monitoring Console (CMC) dashboards to determine if any searches have performance issues that need attention. The CMC enables you to monitor Splunk Cloud Platform deployment health and to enable platform alerts. You can modify existing alerts or create new ones. You can interpret results in these dashboards to identify ways to optimize and troubleshoot your deployment.

- Search usage statistics. This dashboard shows search activity across your deployment with detailed information broken down by instance.
- Scheduler activity. This dashboard shows Information about scheduled search jobs (reports) and you can configure the priority of scheduled reports.
- Forwarders: Instance and Forwarders: Deployment. These dashboards show information about forwarder connections and status. Read about how to troubleshoot forwarder/receiver connection in Forwarding Data.

Improve your searches

- Select an index in the first line of your search. The computational effort of a search is greatest at the beginning, so searching across all indexes (index=*) slows down a search significantly.
- Use the TERM directive. Major breakers, such as a comma or quotation mark, split your search terms, increasing the number of false positives. For example, searching for average=0.9* searches for 0 and 9*. Searching for TERM(average=0.9*) searches for average=0.9*. If you aren't sure what terms exist in your logs, you can use the walklex command (available in version 7.3 and higher) to inspect the logs. You can use the TERM directive when searching raw data or when using the tstats command.
- Use the tstats command. The tstats command performs statistical queries on indexed fields, so it's much faster than searching raw data. The limitation is that because it requires indexed fields, you can't use it to search some data. However, if you are on 8.0 or higher, you can use the PREFIX directive instead of the TERM directive to process data that has not been indexed while using the tstats command. PREFIX matches a common string that precedes a certain value type.
- Avoid using table commands in the middle of searches and instead, place them at the end. Table is a reporting command and will cause data to be pushed to the Search Head which then performs the work, when it's usually more efficient to have the search load distributed among the indexers since they can take advantage of Map Reduce, for example.
- Test your search string performance. The Search Performance Evaluator dashboard allows you to evaluate your search strings on key metrics, such as run duration (faster is better), the percentage of buckets eliminated from a
search (bigger is better), and the percentage of events dropped by schema on the fly (lower is better).

Detect and resolve data imbalances

Run the following search to detect a data imbalance. Specify a time window of 15 minutes or less before running the search.

```
  | tstats count WHERE index=_internal BY splunk_server
```

This search counts the distribution of events across indexers. There are two important things you should look for in the results:

- All your indexers should be listed. If an indexer is missing from the list, there is either a problem distributing the search to that peer (you’d see a warning message) or no events are being sent to that peer.
- An even or close-to-even distribution of events across the peers.

If there is an imbalance of events across the peers, you should correct it as soon as possible. The Splunk UF provides a built-in load balancing mechanism that is enabled by default. However, it may require adjustments for some data sources. The UF is designed to stream data sources to indexers as quickly as possible. Due to its lightweight nature, the UF does not see event boundaries in your log files and data streams. To ensure that events aren’t chopped in half when switching between indexers, the UF waits until it has read to the end of a log file or until a data stream has gone quite before streaming data to a new indexer. This can create issues if the UF is reading from a very large log file or a very chatty data stream. There are two choices for resolving situations where the UF becomes “sticky” to one of more indexers due the conditions discussed above.

- The first is a parameter called ‘forceTimeBasedAutoLB’. This parameter is convenient because it is available in older versions of Splunk and it applies to all sources/sourcetypes handled by a UF. When enabled, this setting causes the UF to switch indexers whenever ‘autoLBfrequency’ or ‘autoLBvolume’ are reached, even if it is still reading from a log or receiving a data stream. This can cause issues if any single event is larger than 64KB when reading a log file or 8KB for an incoming TCP/UDP stream. If any event exceeds those sizes, you run the risk of the event being truncated or lost.
- The second parameter is called ‘event_breaker’. This parameter is enabled on a per-sourcetype basis on each UF. The advantage it has over ‘forceTimeBaseAutoLB’ is that there are no event size limitations. However, this parameter requires you to manage additional configurations on each UF and is only available on forwarders running Splunk 6.5 or newer.

Adjust the search mode

The Splunk Search & Reporting app has multiple modes that searches can be run under. These different search modes impact the resource utilization, search runtime, and data transferred to fulfill your search.

- Fast mode. Searches run in fast mode only return fields that are explicitly mentioned in the search or are indexed-fields. This mode has the lowest impact on system resources, but also limits your ability to visually explore your data via the UI.

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.
• Smart mode. When running searches in smart mode, Splunk attempts to decide which fields are necessary to fulfill your search. For example, if you use a transforming command like ‘stats’ in smart mode, Splunk only returns the summary data and not the raw event.

• Verbose mode. Sometimes it is necessary to see all of the fields and raw data in your search, even if you’ve used a transforming command like timechart. For example, if you’re trying to troubleshoot your search to determine why a graph looks a particular way, you might want to see the raw events. Under smart mode, the raw events won’t be available under the Events tab. You have to switch to verbose mode to see the raw events and the summary data. This mode is extremely inefficient because it causes the indexers to send all events matching your search criteria to the search head. You should only enable verbose mode temporarily to troubleshoot searches.

Target your search to a narrow dataset

Reducing the scope of your search to a more narrow set of results is often the quickest and easiest way to improve performance and reclaim system capacity.

• Limit the timeframe of your search to 15 minutes or less.
• Reduce the amount of data Splunk needs to search through by specifying specific index names in your searches. Typically, you want to store like data that is commonly searched together in the same index.
  ◦ For example, let’s say you have 5 different firewall vendors sending data to Splunk. Even though the data format and sourcetypes are different, you probably write searches that target all firewall data at the same time. Keeping the sourcetypes in the same index prevents Splunk from needing to look in different places to match search terms.
  ◦ For another example, let’s say your firewall data has 5 billion unique events and is stored in the ‘main’ index. You decide to add the error logs from your WordPress site, which is 100k unique events to the same index. Anytime you want to run a search for your WordPress logs, Splunk has to sort through 5 billion firewall events to find the ones you care about. If you moved your WordPress logs to a different index, you could speed up searches by reducing the amount of data Splunk has to sort through.
• Add more unique terms to your search. When running a search, Splunk consults the TSIDX to locate all events that contain the terms provided in your search. For example, consider the following search: index=firewall status=ERROR. Splunk would consult the TSIDX files for the ‘firewall’ index and locate all events that contain the term ‘error’. It is highly likely that many events contain the term ‘error’ and Splunk will need to sort through a lot of data to locate all of those events. You could speed up this search by always specifying terms that are unique to the events you want to target. This search would perform better: index=firewall status=ERROR type=cisco model=asa datacenter=newyork

Take additional steps

• Improve your source types. Review the data quality dashboards to identify and resolve data quality issues.
• Check the HTTP Event Collection status. If you have set up the HTTP Event Collector, you can use it to monitor the progress of a token.
• Use tokens to build high-performance dashboards. Searches saved in dashboards can use tokens to allow users to switch between commands. When the token is in a child search, only the child search is updated as the token input changes. The base search, which can contain the index and other costly functionality, only needs to run once, which speeds up the search overall.
• Preload expensive datasets using loadjob. The loadjob command uses the results of a previous search. If you run a
lengthy search in one browser tab and keep it open, the data remains on the search head for some time, as long as you keep the tab open. Eventually, the search will time out, but while it is available, you can run other searches based on that initial data using the search job id (sid).

- Use **horizontal scaling** to increase concurrency and data ingest rates. Splunk decreases search runtime by dividing up the processing across multiple servers. By doing this, each server performs less overall work, which decreases individual search runtime and increases the number of searches that can be executed in the same span of time. For example, if a search takes 60 seconds to complete on a single server, you can divide that work across 6 servers and complete the same search in 10 seconds.

- Change **scheduler limits**. A Splunk administrator can define what percentage of the total search capacity the scheduler is allowed to consume with scheduled search jobs. By default, the scheduler is allowed to consume 50 percent of the total capacity. This ensures that there is reserved capacity for interactive users to create ad-hoc searches. If you have a high number of scheduled searches, you may choose to raise the scheduler limits.

- **Knowledge objects** can impact the work required and system resources needed to fulfill a search. For example, if you’ve configured an automatic lookup and scoped it for all users globally, Splunk needs to enrich all events in every search with that lookup whether the user needs the additional fields or not. You can avoid unnecessary resource consumption by only installing apps and technology add-ons (TAs) in production that are necessary. After installing, ensure that the App/TA is scoped so that it only targets the appropriate users and searches.

---

### Next steps

These additional Splunk resources might help you understand and implement these recommendations:

- .conf Talk: **How to get the most out of your lexicon**
- .conf Talk: **Master search speed**