Splunk® Enterprise Search Manual 8.2.6

About search optimization

Generated: 9/20/2023 4:57 am

About search optimization

Search optimization is a technique for making your search run as efficiently as possible.

When not optimized, a search often runs longer, retrieves larger amounts of data from the indexes than is needed, and inefficiently uses more memory and network resources. Multiply these issues by hundreds or thousands of searches and the end result is a slow or sluggish system.

There are a set of basic principles that you can follow to optimize your searches.

- Retrieve only the required data
- Move as little data as possible
- Parallelize as much work as possible
- Set appropriate time windows

To implement the search optimization principles, use the following techniques.

- Filter as much as possible in the initial search
- Perform joins and lookups on only the required data
- Perform evaluations on the minimum number of events possible
- Move commands that bring data to the search head as late as possible in your search criteria

Indexes and searches

When you run a search, the Splunk software uses the information in the index files to identify which events to retrieve from disk. The smaller the number of events to retrieve from disk, the faster the search runs.

How you construct your search has a significant impact on the number of events retrieved from disk.

When data is indexed, the data is processed into events based on time. The processed data consists of several files:

- The raw data in compressed form (**rawdata**)
- The indexes that point to the raw data (**index files**, also referred to as **tsidx files**)
- Some metadata files

These files are written to disk and reside in sets of directories, organized by age, called **buckets**.

Use indexes effectively

One method to limit the data that is pulled off from disk is to partition data into separate indexes. If you rarely search across more than one type of data at a time, partition different types of data into separate indexes. Then restrict your searches to the specific index. For example, store web access data in one index and firewall data in another. Using separate indexes is recommended for sparse data, which might otherwise be buried in a large volume of unrelated data.

- See Ways to set up multiple indexes in the *Managing Indexers and Clusters of Indexers* manual
- See Retrieve events from indexes

A tale of two searches

Some frequently used searches unnecessarily consume a significant amount of system resources. You will learn how optimizing just one search can save significant system resources.

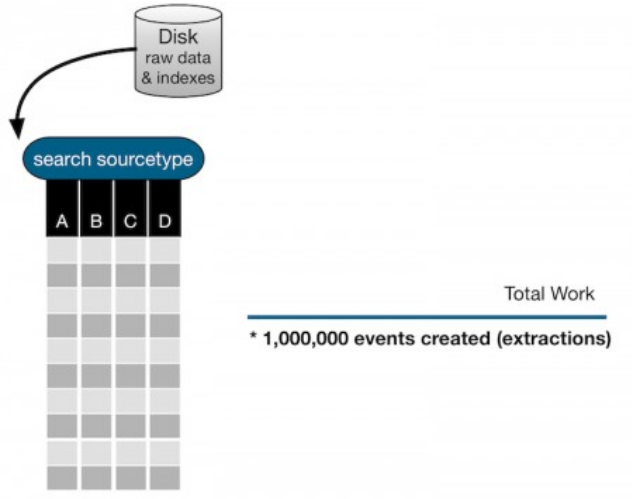
A frequently used search

One search that is frequently used is a search that contains a lookup and an evaluation, followed by another search. For example:

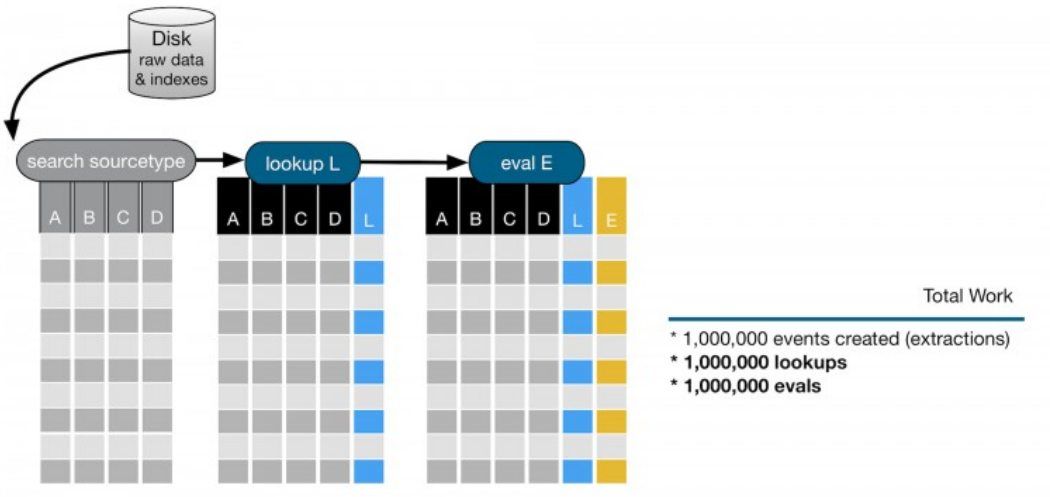
```
sourcetype=my_source | lookup my_lookup_file D OUTPUTNEW L | eval E=L/T | search A=25 L>100 E>50
```

The following diagram shows a simplified, visual representation of this search.

When the search is run, the index is accessed and 1 million events are extracted based on the source type.



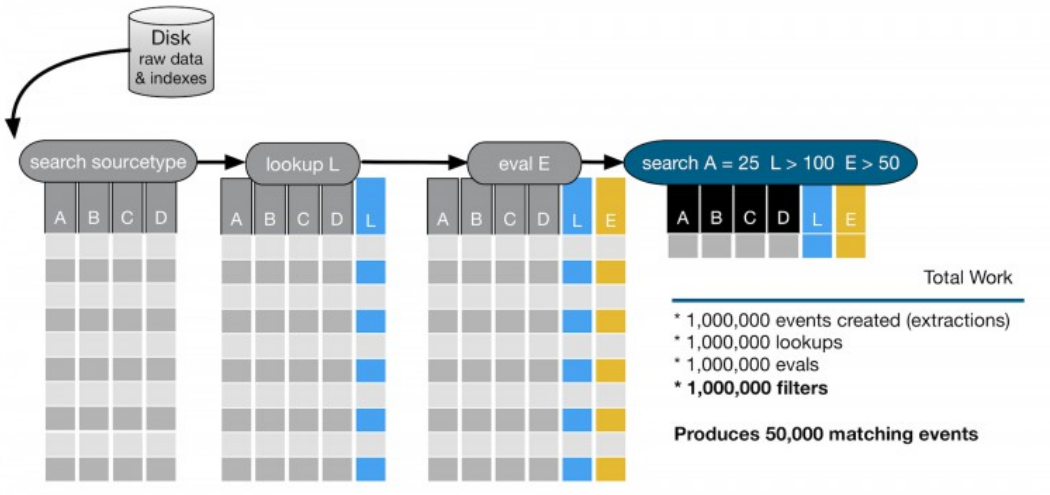
In the next part of the search, the `lookup` and `eval` command are run are on all 1 million events. Both the `lookup` and `eval` commands add columns to the events, as shown in the following image.



Finally, a second search command runs against the columns A, L, and E.

- For column A, the search looks for values that are equal to 25.
- For column L, which was added as a result of the `lookup` command, the search looks for values greater than 100.
- For column E, which was added as a result of the `eval` command, the search looks for values that are greater than 50.

Events that match the criteria for columns A, L, and E are identified, and 50,000 events that match the search criteria are returned. The following image shows the entire process and the resource costs involved in this inefficient search.



An optimized search

You can optimize the entire search by moving some of the components from the second `search` to locations earlier in the search process.

Moving the criteria `A=25` before the first pipe filters the events earlier and reduces the amount of times that the index is accessed. The number of events extracted is 300,000. This is a reduction of 700,000 compared to the original search. The `lookup` is performed on 300,000 events instead of 1 million events.

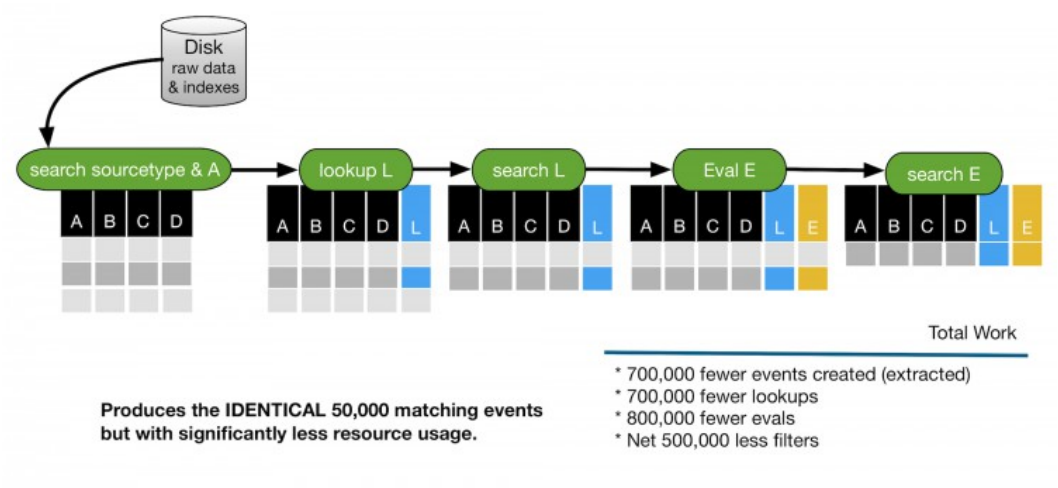
Moving the criteria `L>100` immediately after the `lookup` filters the events further, reducing the number of events returned by 100,000. The `eval` is performed on 200,000 events instead of 1 million events.

The criteria `E>50` is dependent on the results of the `eval` command and cannot be moved. The results are the same as the original search. 50,000 events are returned, but with much less impact on resources.

This is the optimized search.

```
sourcetype=my_source A=25 | lookup my_lookup_file D OUTPUTNEW L | search L>100 | eval E=L/T | search E>50
```

The following image shows the impact of rearranging the search criteria.



See also

- Quick tips for optimization
- Write better searches
- Built-in optimizations