Mastering Splunk Searches: Improve searches by 500k+ times

Andrew Landen
Sr Splunk Developer | Chevron

Who is Andrew Landen?

1) **Education**

   Physics, BS

   Information Systems Security, MS

2) **Experience**

   Teaching (3 yrs)

   Geophysics (2 yrs)

   IT Security/Splunk (6 yrs)

3) **Splunk Experience**

   Sr. Developer (4 yrs)

   Splunk Architect, SAE/CSM at Splunk (2 yrs)

   Sr. Splunk Developer with Chevron (present)

Splunk IT Service Intelligence™

Splunk … When all HEC breaks loose - klaxdal

Cool t-shirt ideas from you!

"NOC NOC. Who's there? IT'S I."

landen99

I'll cast a SPL on you. And now() your mine. – landen99

Splunk your data and party like it's 946684740 –mikekramer

https://answers.splunk.com/answers/686727/what-are-your-splunk-t-shirt-ideas.html

Relevant .conf Talks

Useful reference links for this talk

Fields, Indexed Tokens, And You by Martin Müller @ .conf2017

The primary basis for this talk

Optimizing Splunk Knowledge Objects by Martin Müller @ .conf2015

Beware of "Unintended Consequences"!

Lesser Known Search Commands by Kyle Smith @ .conf2016

Cool commands, like multisearch, to increase your arsenal of SPL tools

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only, and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Turn Data Into Doing, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2019 Splunk Inc. All rights reserved.

Agenda and Objectives

What are we going to talk about?

Agenda

What are we going to talk about?

Search Process Overview

Fast Search Types

Segmentation/breakers: Major and minor

Subsearch SPL filter generation

Tstats

Multisearch

Objective

What kind of results can I expect?

Slow stats search

After 86,271.264s: This search is still running and is approximately 3.221% complete.

Fast TERM-stats search

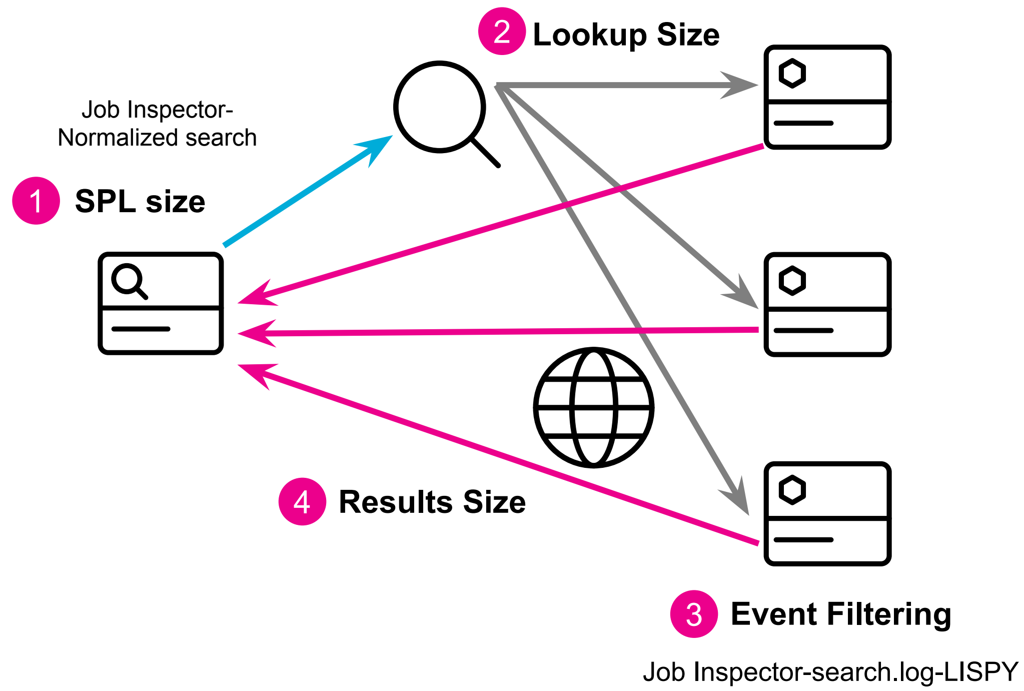
This search has completed and has returned 6 results by scanning 10,000,000,000 events in 0.13 seconds

Tip: Insane search speeds with tstats-TERM searches

Search Process Overview

Which part of the search is taking the longest?

1. SPL size
   - Indexed Inspect the normalized search
   - Subsearch: 10k/50k, 60s
2. Lookup size
3. Event filtering
   - Indexed fields/TERM
4. Results size
   _raw vs summary table



Fast Search Types

7 sub-categories of fast searches

TERM

- index=a1 TERM(f1=v1)
- | tstats count where index=a1 TERM(f1=v1) by _time span=1d
  - TERM applies to raw, not datamodels: from DM.DM

Summary data

- index=summary | collect index=a1 testmode=t
- | loadjob SID
- | inputlookup a1.csv where f1=v1

System calls

- | metadata index=a1 type=hosts
- | rest /services/saved/searches/

Segmentation

Indexed token event filtration

Segmentation and Segmentors

How does segmentation work?

Breakers are defined in Segmentors.conf:

- Major: [ ] < > ( ) { } | ! ; , ' " * \n \r \s \t & ? + %21 %26 %2526 %3B %7C %20 %2B %3D %2520 %5D %5B %3A %0A %2C %28 %29
- Minor: / : = @ . - $ # % \ _

Segmentation Example:

Spaces are major breakers

- [24/Oct/2019:09:11:01.404 -0500] src=127.0.0.1;50
- Ex: Find all events with a src ip of 127.0.0.1

  index=a1 TERM(2019) TERM(src=127.0.0.*) TERM(50)

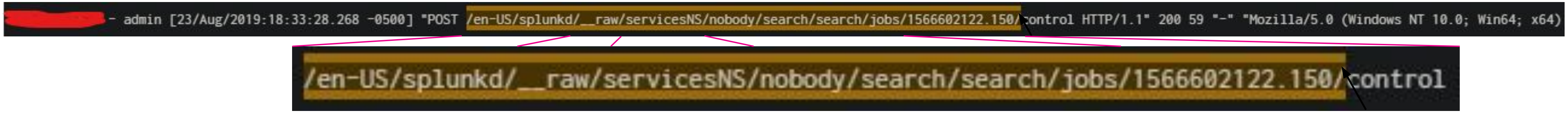
Not case sensitive

"*" – SPL wildcard/segment major breaker

Exploring Segmentation

Splunk GUI highlights segmentation with mouseover

Splunk GUI Highlights TERM

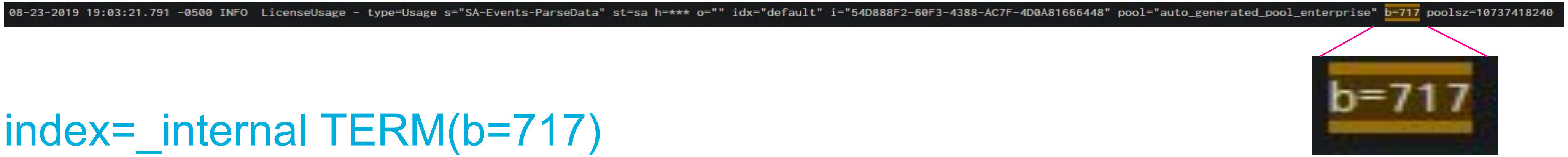


```
[redacted] - admin [23/Aug/2019:18:33:28.268 -0500] "POST /en-US/splunkd/__raw/servicesNS/nobody/search/search/jobs/1566602122.150/control HTTP/1.1" 200 59 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

```
/en-US/splunkd/__raw/servicesNS/nobody/search/search/jobs/1566602122.150/control
```

index=_internal TERM(/en-US/splunkd/__raw/servicesNS/nobody/search/search/jobs/1566602122.150/*)

index=_internal 1566602122.150



```
08-23-2019 19:03:21.791 -0500 INFO  LicenseUsage - type=Usage s="SA-Events-ParseData" st=sa h=*** o="" idx="default" i="54D888F2-60F3-4388-AC7F-4D0A81666448" pool="auto_generated_pool_enterprise" b=717 poolsz=10737418240
```

```
b=717
```

index=_internal TERM(b=717)

TERM

Unique indexed values are the key to speed

> “In this segment,
> we’ll come to TERMs with
> the speed of major and
> minor breakers.”

Splunk Search Master

Filtering on Major Segments

Event matching with TERM

To select events certain usernames like: timestamp=x username=user1 foo=bar

- TERM(username=user1) OR TERM(username=user2) OR ..
- Naturally excludes: username=-
- Much faster than: (username=user1 OR username=user2 OR …)
- Unintended matches: url="bad.com/ohno;username=user1;oops"
- Avoid early wildcards like: TERM(*foo*bar)

Best log format: field=value

- Avoid major breakers in field values; they break the TERM key-value pair
  x x
  - foo,username="user1 user2",bar=val
  x x
  - foo,category=v1a v1b;v2,bar=val

Filtering Out with "NOT TERM"

Filtering out events with TERM

To ignore events without username like: timestamp=x username=- other=data

- Use: NOT TERM(username=-)
- Much faster than: NOT username=- OR username!=-
- Unintended filter matches: url="bad.com/ohno;username=-;oops"
- Avoid early wildcards like: NOT TERM(*foo*bar)

Construct SPL with Return Command

Custom TERM SPL filters from any result set

Usage:

Mysearch .. | stats count by foo

- | eval bar=TERM(".foo."*)" | return 9 $bar
  - Yields SPL: TERM(foo1*) OR TERM(foo2*) OR ..
  - TERM is unnecessary without minor breakers: foo1* OR foo2* OR ..
  - TERM breaks with major breakers: TERM("foo1"1*) OR TERM(foo1;2*) OR ..
- To add a value prefix like "src=": | eval foo=TERM(src=".foo."*)" | return 9 $foo
  - Yields SPL: TERM(src=foo1*) OR TERM(src=foo2*) OR ..

Demo

TERM filter with Time Window

Filtering with dynamic TERM and time windows

Create the SPL:

```
mysearch .. | stats count by foo _time
| eval earliest=_time-10*60,latest=_time+10*60,
bar="TERM(".foo."*) earliest=".earliest." latest=".latest | return 9 $bar
```

- Yields SPL: (TERM(foo1*) earliest=x latest=y) OR ..

Temporal filter options:

- (earliest=-1d latest=now) OR (_time>[epoch1] _time<[epoch2])
- (_index_earliest=-h@h _index_latest=@h) OR (_indextime>[epoch1] _indextime<[epoch2])
- [epoch1], [epoch2] – epoch temporal window (no timezone)

Demo

TERM macro filter

Searching TERM list values with an easy macro

```
| inputlookup b | `term("b=",b)`
```

```
term(2): stats count by $f$ | rename $f$ AS f | table f | eval f = "TERM($pre$".f.")" | return 999999 $f
```

Yields: (TERM(b=140122)) OR (TERM(b=143)) OR (TERM(b=3037)) OR (TERM(b=717)) OR (TERM(b=771)) OR (TERM(b=916))

```
index=_internal [| inputlookup b | `term("b=",b)`]
```

Yields: index=_internal (TERM(b=140122)) OR (TERM(b=143)) OR (TERM(b=3037)) OR (TERM(b=717)) OR (TERM(b=771)) OR (TERM(b=916))

Values macro filter

Searching list values with an easy macro

```
| inputlookup b | `values(b)`
  values(1): stats count by $f$ | rename $f$ AS f | table f | return 999999 $f
    Yields: (140122) OR (143) OR (3037) OR (717) OR (771) OR (916)
```

```
index=_internal [| inputlookup b | `values(b)`]
    Yields: index=_internal (140122) OR (143) OR (3037) OR (717) OR (771) OR (916)
```

Multisearch and Crossjoins

Dynamic searches using multiple data sources

Multisearch with Dynamic Filter

Splunk shines with correlating multiple datastreams

```
| multisearch

[ search index=a TERM(src=8.8.8.8)]

[ search index=b
    [search index=a TERM(src=8.8.8.8) | eval search="TERM(dest=".dest.")" | return 9 $search] ]
```

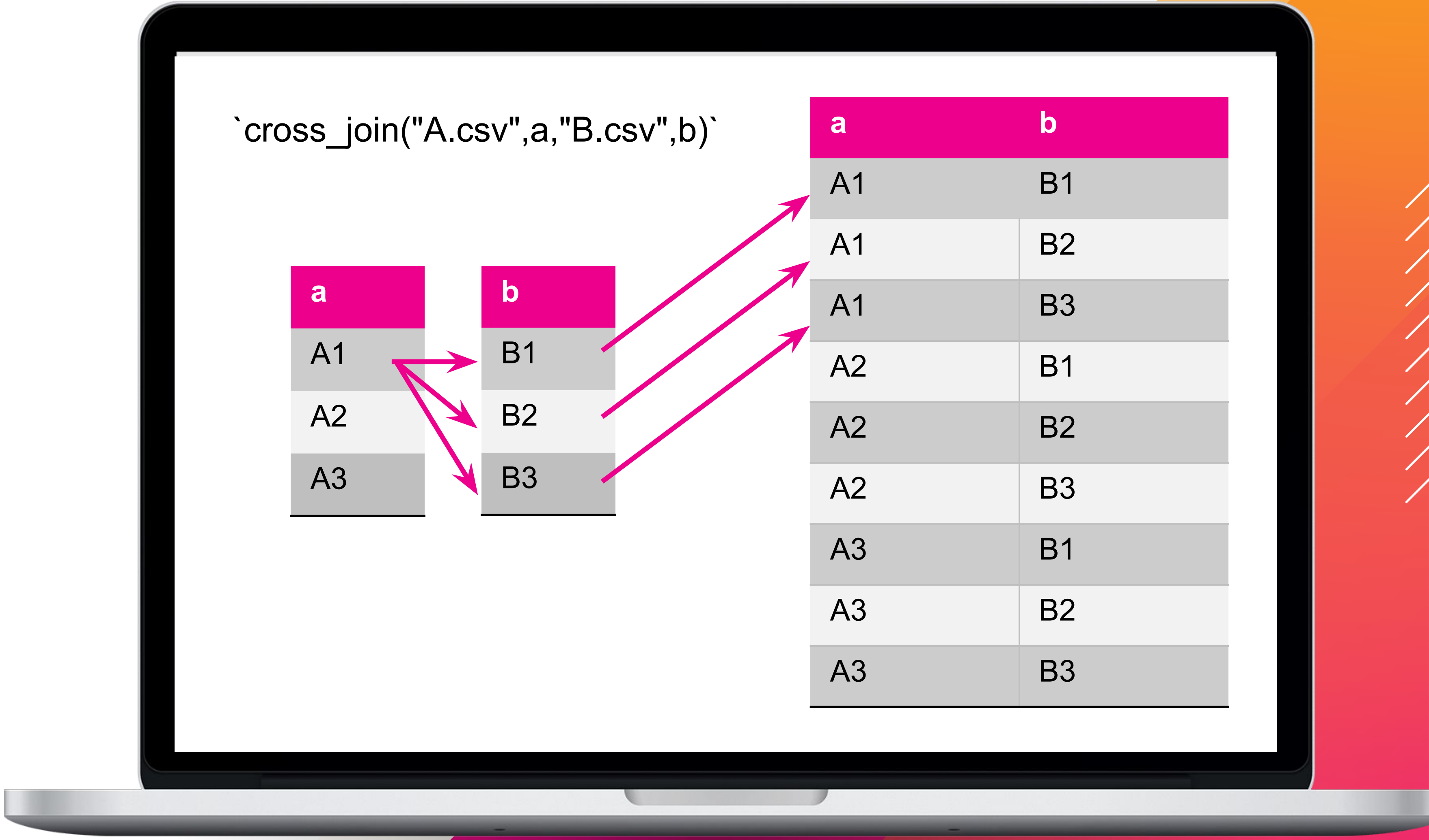
Multisearch allows decentralized streaming commands like: eval, rex, fields

Crossjoin Lists A and B

Stats: the fast join/crosscjoin

cross_join

```
| inputlookup A.csv
| inputlookup append=t B.csv
| stats values(a) values(b)
| rename values(*) AS *
| stats count by a b | table a b
```



```
`cross_join("A.csv",a,"B.csv",b)`
```

a
A1
A2
A3

b
B1
B2
B3

a	b
A1	B1
A1	B2
A1	B3
A2	B1
A2	B2
A2	B3
A3	B1
A3	B2
A3	B3

Q&A

"The answer is only as good as the question."
Andrew Landen | Sr. Splunk Dev @ Chevron

Key Takeaways

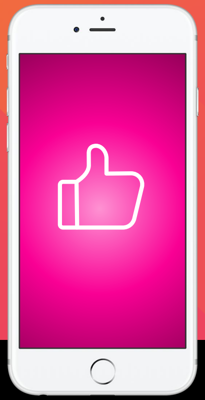
Speed rests on the parallel indexers, so use them wisely.

1. Effective event selection is the primary method for improving search speeds.
2. Indexed fields and tokens, including time, are the fastest event filters.
3. Any search can create TERM filters to greatly accelerate any other search.

.conf19

splunk>

Thank You!



Go to the .conf19 mobile app to

RATE THIS SESSION