splunk> .conf2017

Search Performance Improvements

What we’ve done and why we did it…

Alex James – Senior Principal Product Manager (Search Technologies)
Manan Brahmkshatriya - Principal QA Engineer

25-28th Sept 2017 | Washington, DC

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance 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 could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. 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 functionality described or to include any such feature or functionality in a future release.

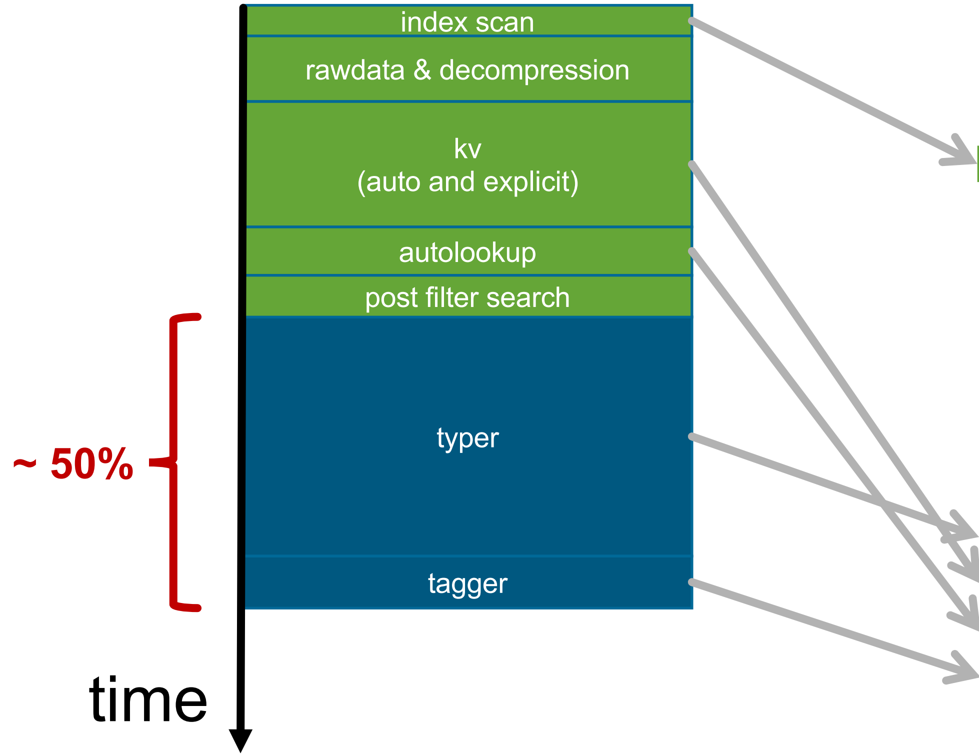
Session Outline

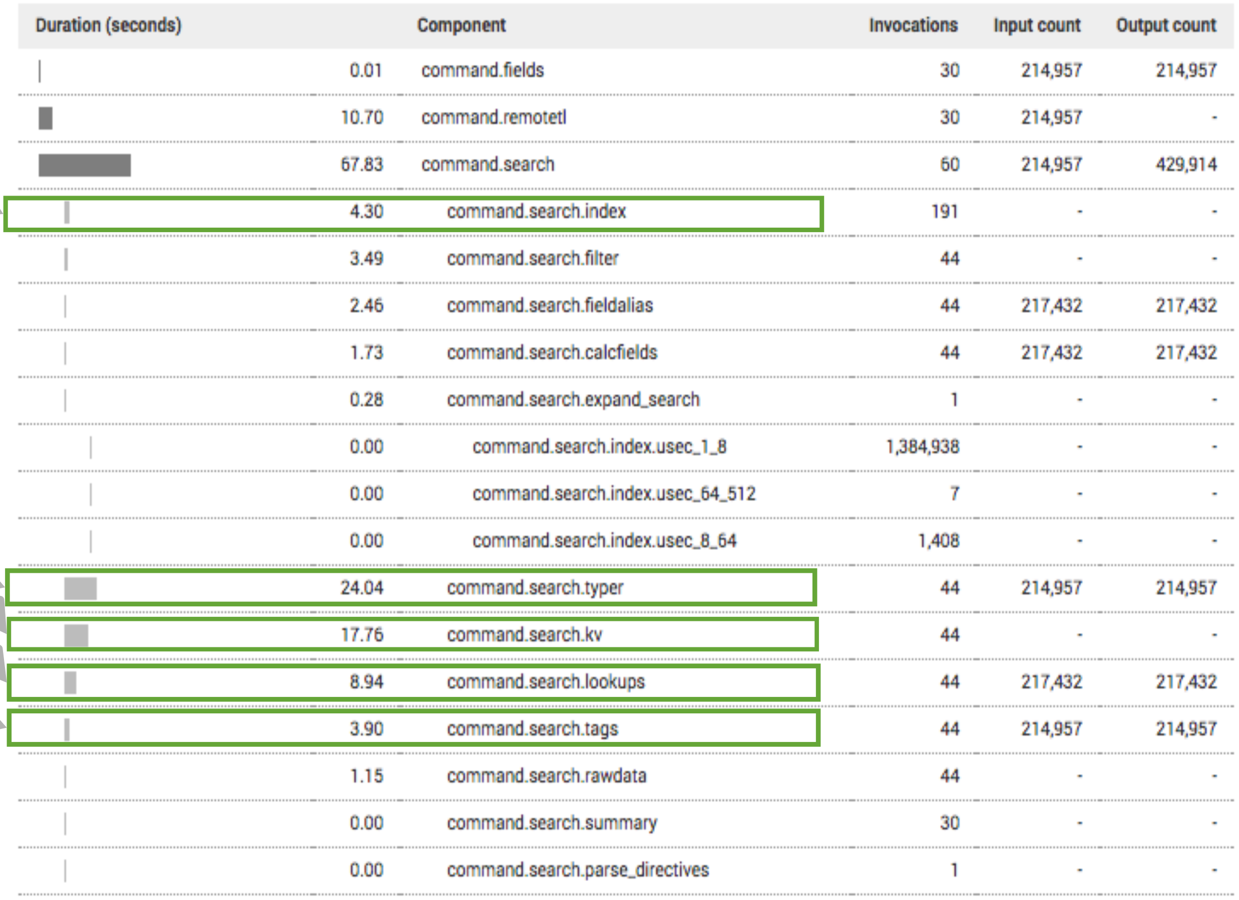
- Language Improvements
- Data Model Improvements
- Optimizer Improvements
- Further Improvement Ideas
- Q&A

SPL Language Improvements

Generating Search – typical breakdown

i.e. the time taken for the first search processor to do its job, with lots of TAs.





Duration (seconds)	Component	Invocations	Input count	Output count
0.01	command.fields	30	214,957	214,957
10.70	command.remotetl	30	214,957	-
67.83	command.search	60	214,957	429,914
4.30	command.search.index	191	-	-
3.49	command.search.filter	44	-	-
2.46	command.search.fieldalias	44	217,432	217,432
1.73	command.search.calcfields	44	217,432	217,432
0.28	command.search.expand_search	1	-	-
0.00	command.search.index.usec_1_8	1,384,938	-	-
0.00	command.search.index.usec_64_512	7	-	-
0.00	command.search.index.usec_8_64	1,408	-	-
24.04	command.search.typer	44	214,957	214,957
17.76	command.search.kv	44	-	-
8.94	command.search.lookups	44	217,432	217,432
3.90	command.search.tags	44	214,957	214,957
1.15	command.search.rawdata	44	-	-
0.00	command.search.summary	30	-	-
0.00	command.search.parse_directives	1	-	-

Search Directives

- Producing TAGS & EVENT TYPES is very costly
  - With lots of TAs it can easily be 50% of the total cost of the search
  - Tags are stored in one multi-valued field
  - We treat as ALL or NOTHING
- Now have a way to selectively request just one or more TAGS (and types)

```
search 500 DIRECTIVES(REQUIRED_TAGS(tags="foo, bar"))
```

```
search 500 DIRECTIVES(REQUIRED_EVENTTYPES(eventtypes="alpha,omega"))
```

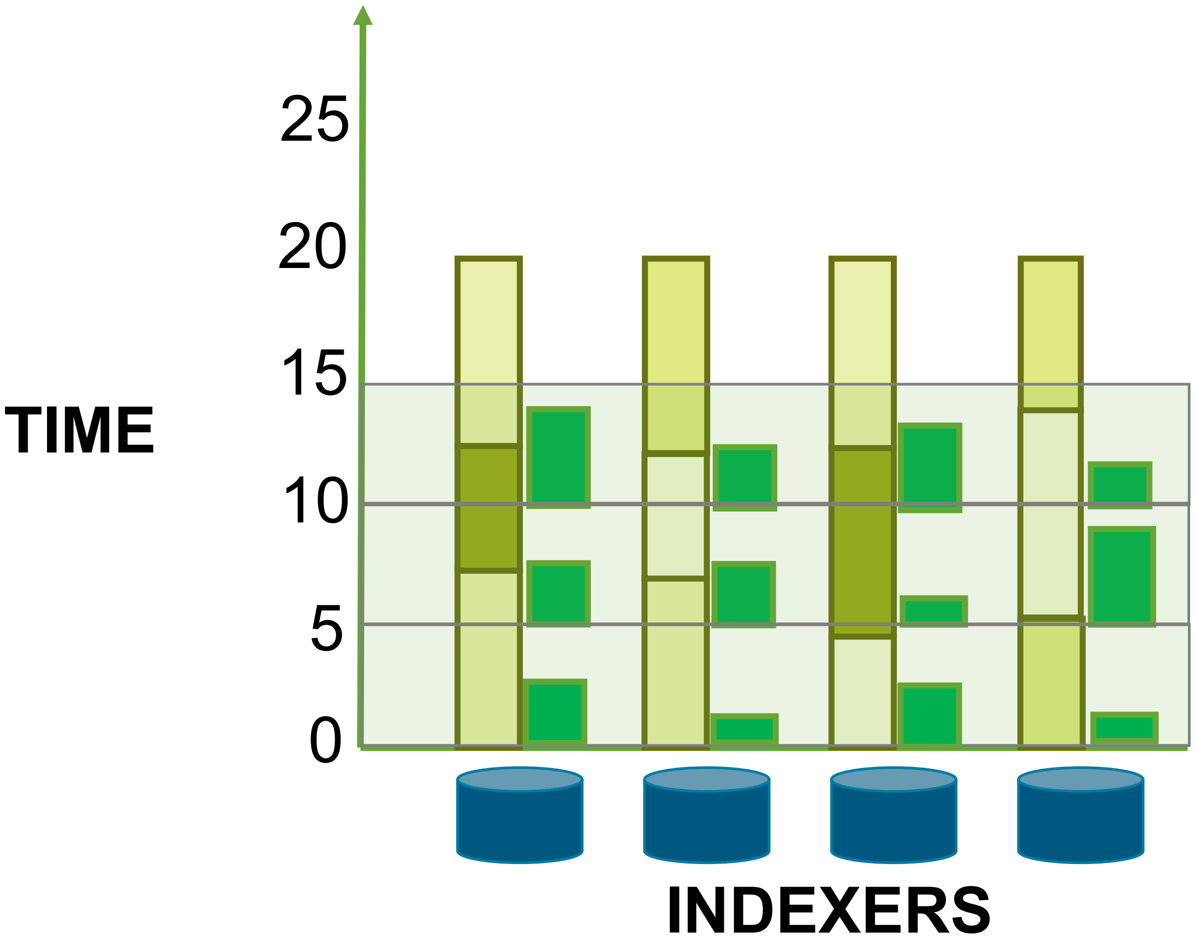
- Combining Directives…

```
search 500 DIRECTIVES(REQUIRED_EVENTTYPES(eventtypes="alpha,omega"),REQUIRED_TAGS(tags="foo,bar"))
```

  - Will produce list of EVENT TYPES needed to correctly produce **foo** and **bar** tags
  - And merge with “**alpha**, **omega**” event types...
- Impact
  - Low – targeted searches for a few events
  - High – broad searches returning lots of events (i.e. Monitoring & Acceleration)

splunk> .conf2017

How Data Model Acceleration works…



Data Model Acceleration (DMA)

Problem and Solution

Issues prior to 7.0:

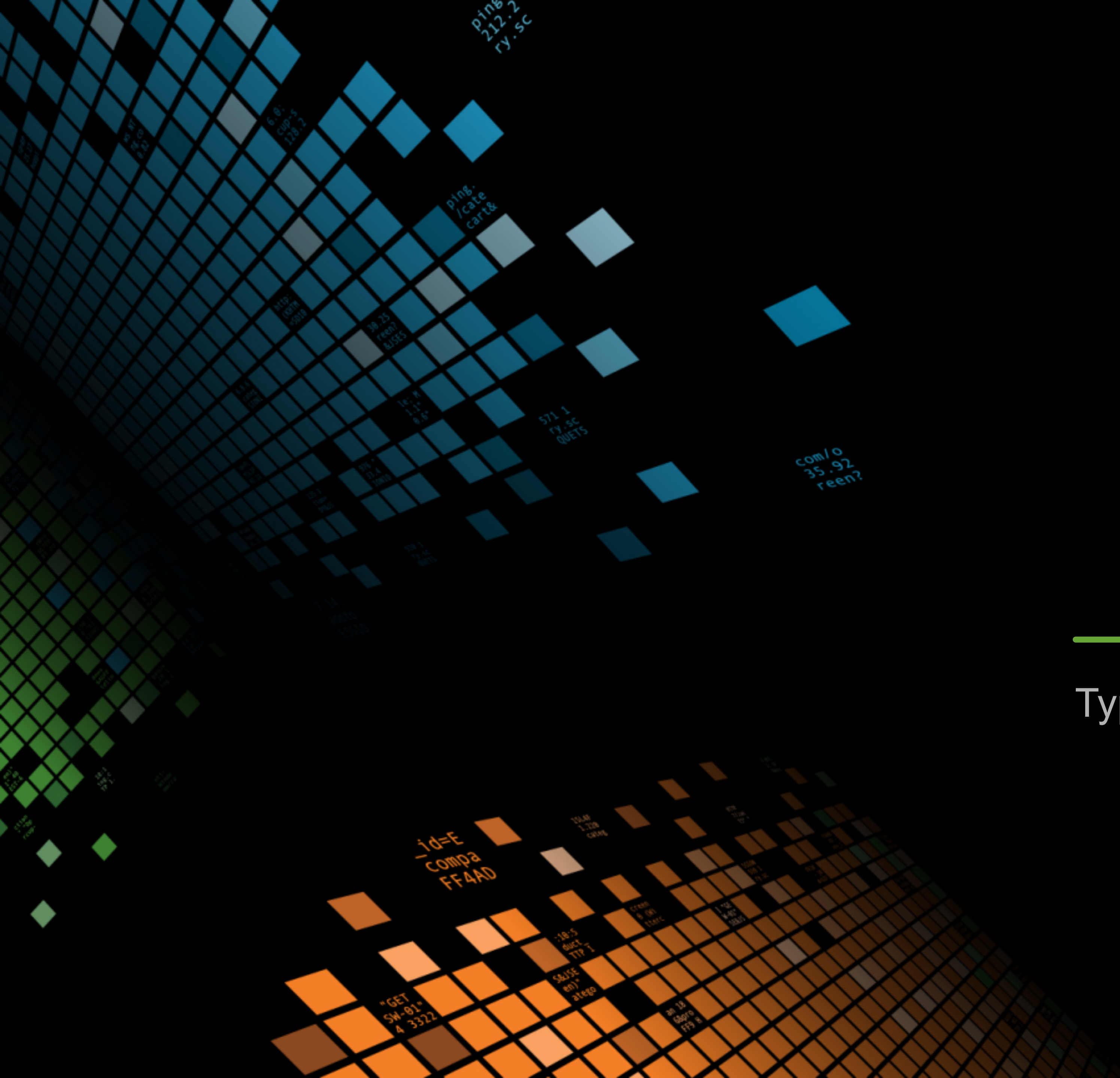
- Acceleration of warm/cold buckets was all or nothing. *(I've started so I'll finish...)*
- So acceleration of a large warm/cold bucket could monopolize acceleration.
- Slowest indexer holds up the other indexers.
- So even temporary data imbalance could lead to loss of parallelism, and cascading delays.

Solution:

- Added ability to pause / continue accelerating warm/cold buckets. *(I've started, but something more important / hot has come along…)*
- This means **acceleration.max_time** is now fully respected, even when processing historical data.
- Next acceleration search starts with hot buckets, thus keeping lag low, even when rebuilding acceleration from scratch.
- If summarization search finishes early we can poll for new data (to reduce lag) so all indexers can be keep busy.
  - See new setting **acceleration.poll_buckets_until_maxtime=true**

Impact:

- 7.0 typically twice as fast as 6.5 (or faster).
- 7.0 lag typically 50% as 6.5 (or less).
- Data Model Acceleration Rebuilds have less impact.



Demo #1

Typer / Tagger and DMA improvements

Improved High Cardinality Processing

Using Parallel Reduce

- Imagine a search like this:
  - search tag=authentication | stats sum(bytes) by host
- Main gate on parallelism / scalability is the number of hosts
- But if we implicitly shuffle before the stats:
  - search tag=authentication | shuffle by host | stats sum(bytes) by host
  - Reduction can happen in parallel
- Limited support for this in 7.0:
  - Needs both:
    - Global enablement (`phased_execution=true` in limits.conf)
    - SPL search by search enablement (`| noop phase_mode=3`)
  - Works with only: **stats**, **transaction** and **tstats**
- Much more coming...

Demo #2

New Optimizations in 7.0

New Optimizations in 7.0

- Projection Elimination for Reporting Commands
  - `search ERROR | eval x=a*b | lookup users uid OUTPUT username | stats count by host`
  - `search ERROR | stats count by host`
- Predicate Splitting
  - `| eval x = a+b | where x=10 and y=10`
  - `| where y=10 | eval x = a+b | where x=10`
- Tag Elimination
  - `search ERROR | where tag=“Authentication” | stats count by host`
  - `search DIRECTIVES(REQUIRED_TAGS(tags=“Authentication”)) | where tag=Authentication | stats count by host`
- Collapsing evals commands
  - `| eval x=a+b | eval y=c+d`
  - `| eval x=a+b, y=c+d`
- Predicate Normalization
  - `search ERROR | where 10=y`
  - `search ERROR y=10`
  - Why would you ever do this:
    - `search ERROR |… |… | eval x=10|… |… | where x=y`

Further Improvement Ideas

splunk> .conf2017

Further Improvement Ideas (1)

- **Faster Lookups and Lookup Replication**
- **Better data structures and serialization formats**
- **More optimization**
  - Projection Elimination for Fields

    -
      ```
      search ERROR | eval x=a*b | inputlookup users uid OUTPUT username | fields b, username
      ```

    -
      ```
      search ERROR | inputlookup users uid OUTPUT username | fields b, username
      ```
  - Merging into Inputlookup (KV Store)

    -
      ```
      | inputlookup foo | search x=10
      ```

    -
      ```
      | inputlookup foo where x=10
      ```
  - Etc.

Further Improvement Ideas (2)

- **Better Parallel Reduce**
  - Implicit support for more reporting commands
  - Better timeliner and preview integration
  - Continued parallel execution (for **both streaming** & **compatible reporting splits**)

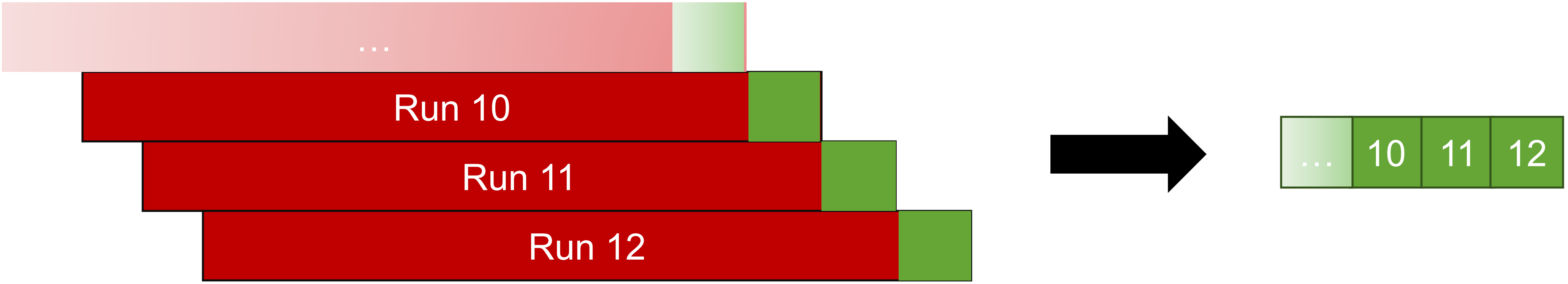
    -
      ```
      | tstats values(Authentication.app) as app, latest(Authentication.user_bunit) as user_bunit from datamodel=Authentication.Authentication by Authentication.user, Authentication.src _time span=1s
      | eventstats dc(Authentication.src) as src_count by Authentication.user
      | search src_count>1
      ```
  - Explicit Shuffle support

    -
      ```
      search tag=authentication | shuffle by host | <any spl>
      ```
- **Better support for result reuse…**

Sliding Window Re-use

Example of Result Reuse

- Lots of searches are scheduled to run on a frequent schedule (every 5m,10m,15m) but cover a larger time range (last 1h, 3h, 24h).
- Which means there is a lot of re-calculation occurring
  - i.e. For a search over the last hour run every 5 mins, ~55mins worth of results have already been calculated once (for the last run) but thrown away.



- Report Acceleration (RA) has the ability to incrementally build results already.
  - Unfortunately RA doesn’t work for TSTATS searches.
  - Why? TSTATS searches leverage Data Model Acceleration (DMA) and we don’t support RA over DMA.
- Many Sliding Windows searches are based on TSTATS
  - Currently investigating adding support for RA over DMA

Summary - What does this mean for you?

- Faster Searches
- Faster Enterprise Security
- Look for opportunities to use new DIRECTIVES
- Checkout the optimizer in the Job Inspector
- Upgrade to 7.0 (or at least 6.5 if that isn’t possible).

Q&A

Alex James - Senior Principal Product Manager

Manan Brahmkshatriya – Principal QA Engineer

Key Takeaways

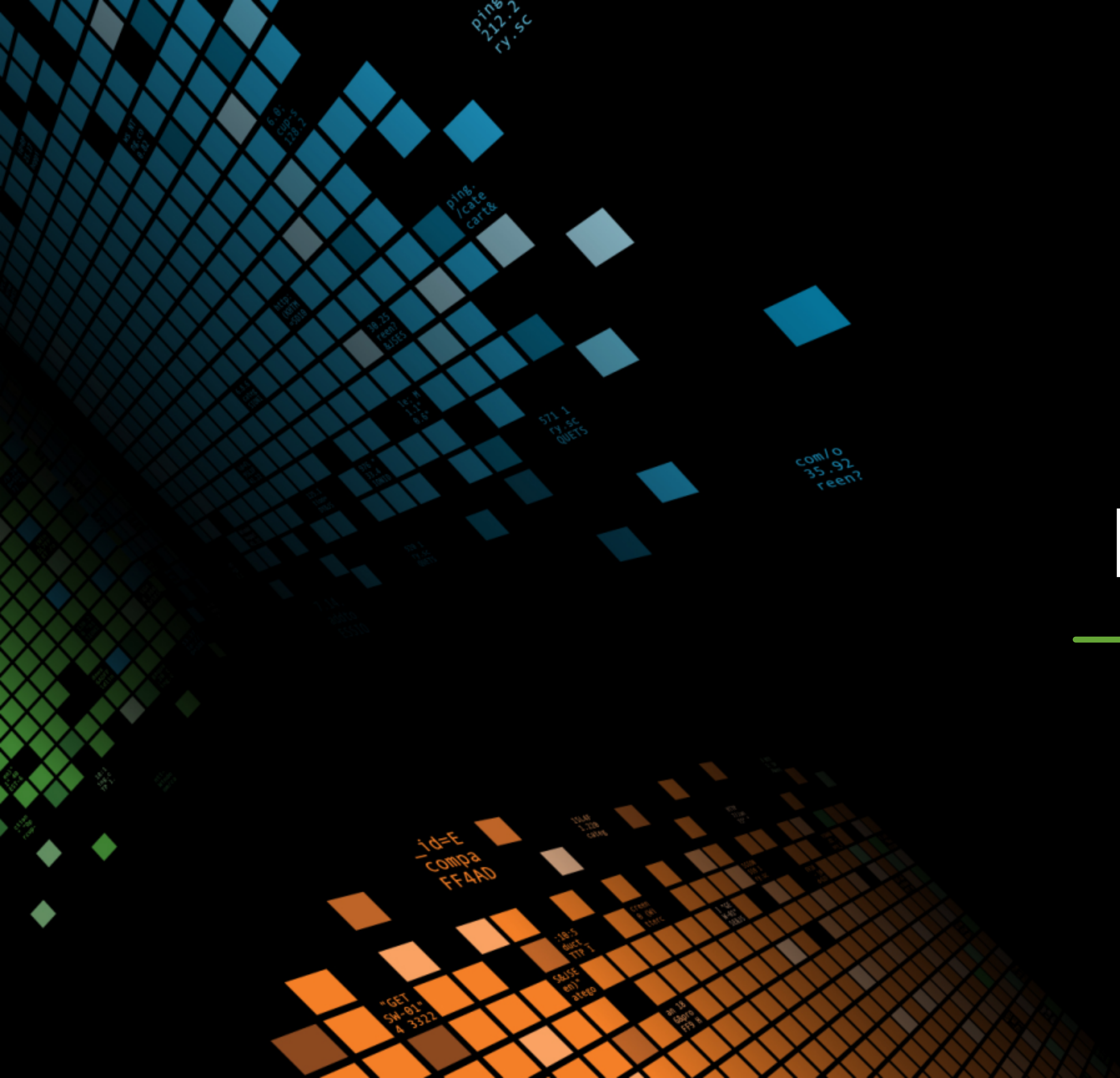
This is where the subtitle goes

1. Splunk 7.0 is significantly faster.
2. Key improvements include: new directives, optimizer improvements and DMA improvements.
3. If you have ES the difference in DMA is very significant.

Thank You

Don't forget to rate this session in the .conf2017 mobile app

splunk> .conf2017



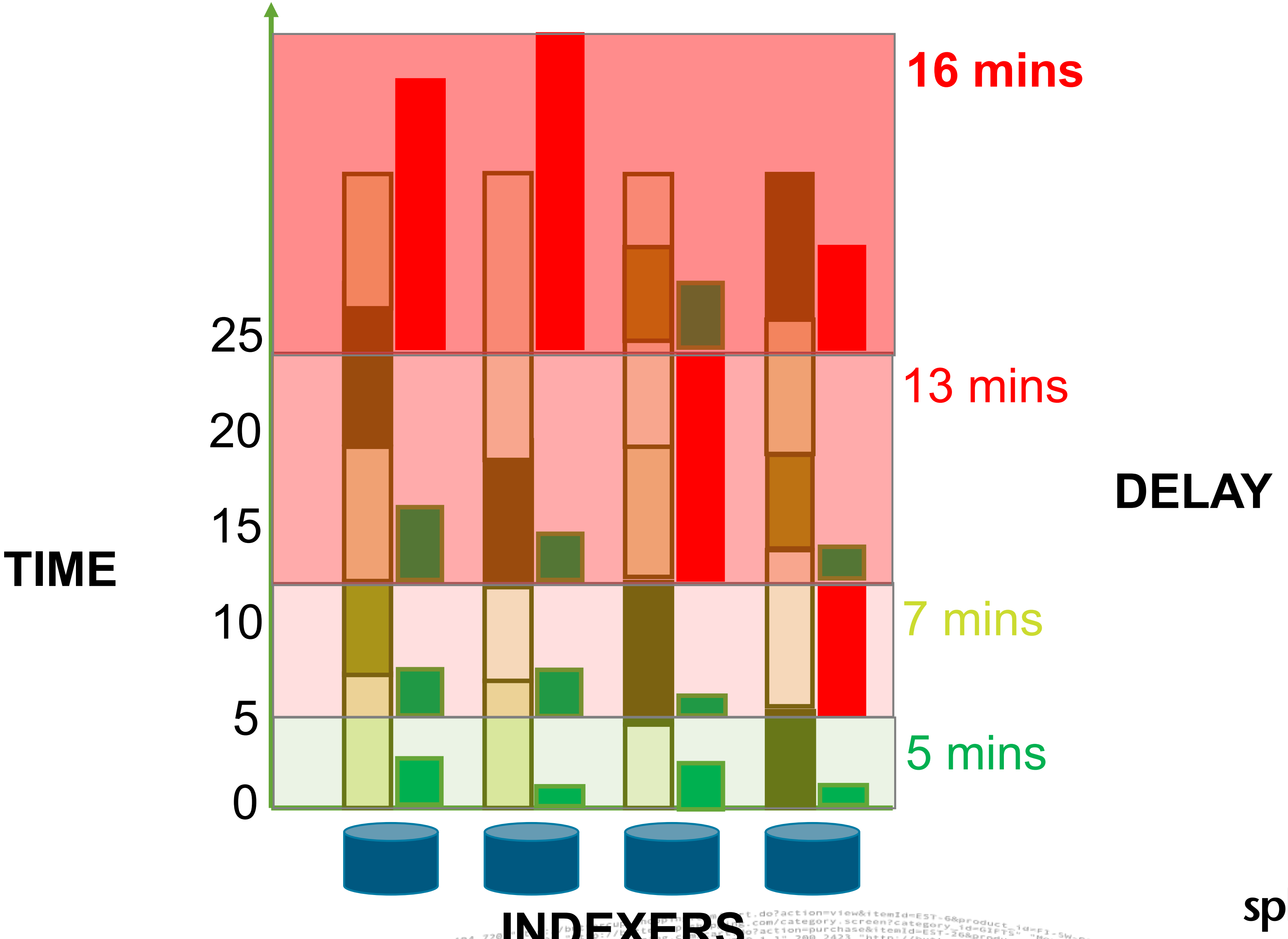
Backup Slides

If the session runs short…

Union

- Similar to **append** but is streaming when possible:
  - `| union [search …| lookup cust id OUTPUT name ], [search …| eval name="SPLK"]`
  - `Returns same data as:`
  - `search …| lookup cust id OUTPUT name | append [search …| eval name="SPLK"]`
- <except> it runs in parallel on indexers (using an improved version of **multisearch** when possible)
- Useful for correlation searches, i.e. append | stats to do a pseudo join
- Supports:
  - More than 2 datasets: **| union [<spl1>], [<spl2>], ... , [<splN>]**
  - Named dataset format (like from) : **| union savedsearch:mysavedsearch, [<spl2>], inputlookup:threats**
  - Shorthand (like append): **<spl1> | union [<spl2>]**
- Should still use a single search or tstats append if possible...
  - Don't do this: **search "error" | union [search "warning"]**
  - Do this: **search "error" OR "warning"**

Effect of temporary data imbalance prior to 7.0


16 mins
13 mins
7 mins
5 mins
DELAY
TIME
INDEXERS