

# The Jiffy Lube Quick Tune-up For Your Splunk Environment

Sean Delaney

Principal Architect, Splunk

Jeff Champagne

Staff Architect, Splunk

.conf2016

splunk >

# Disclaimer

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 the 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.

# What Do These Guys Know Anyway?

## Sean Delaney

[sdelaney@splunk.com](mailto:sdelaney@splunk.com)

Principal Architect

- 5+ Years at Splunk
- Former customer working in the security space
- Former PS consultant
- Leads the Splunk Architecture Council
- Taylor Swift super fan and Hawaiian shirt lover



## Jeff Champagne

[jchampagne@splunk.com](mailto:jchampagne@splunk.com)

Staff Architect

- 2 Years at Splunk
- Former Splunk customer in the Financial Services Industry
- Lived previous lives as a Systems Administrator, Engineer, and Architect
- Loves Skiing, traveling, photography, and a good Sazerac



# Am I In The Right Place?

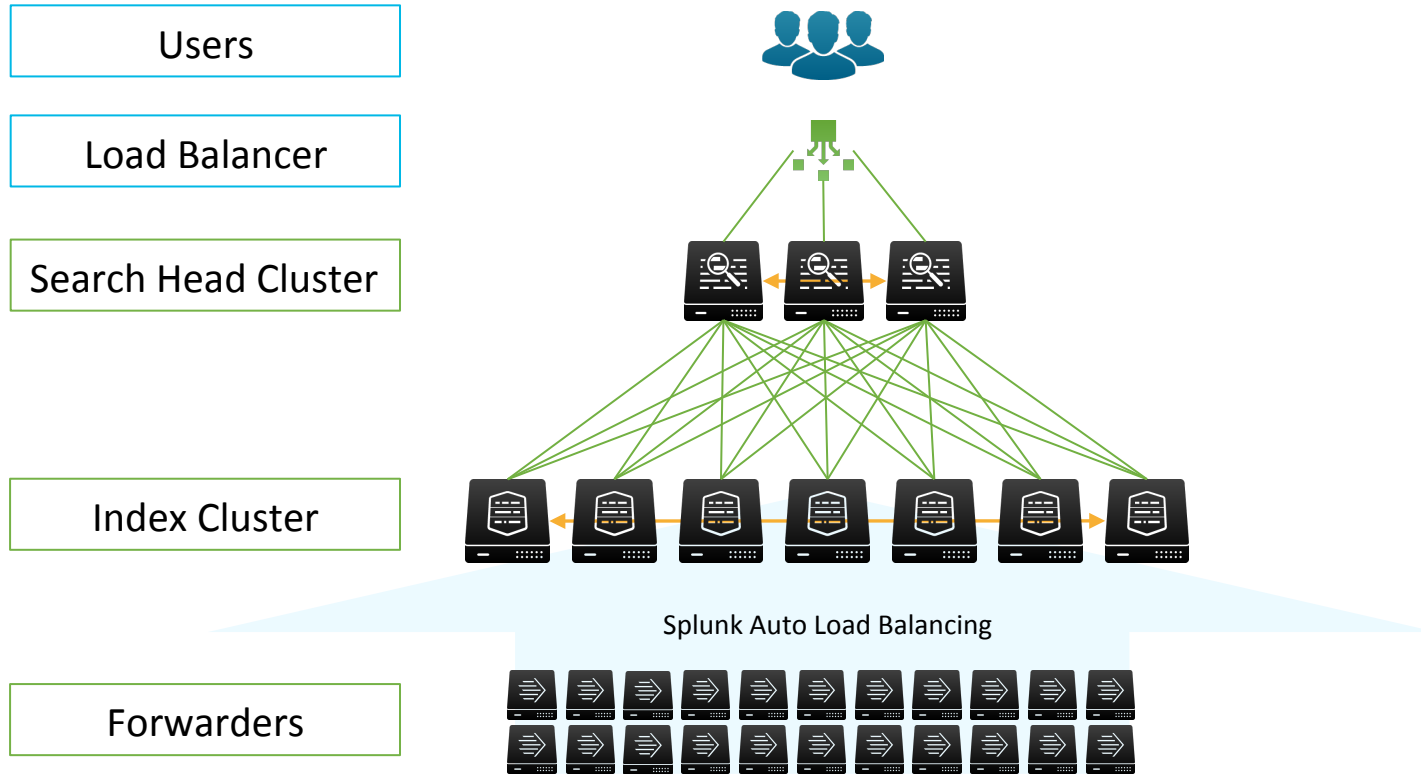
Yes, if you are...

- A beginner/intermediate Splunk Admin
  - Not a total n00b
  - You should already be running Splunk
  - Advanced/Expert Admins, these are not the Droids you're looking for...
- Be familiar with configuration files
- Know what the DMC/Monitoring Console is
- Understand the basic Splunk Distributed Architecture

# What Is A Splunk Tune-up?

- Identify Common Performance Issues
  - Time issues
  - Lagging events
  - Monitoring Indexer Queues
  - Identify slow and resource intensive searches
- Address Performance Issues via Configuration Changes

# Splunk Distributed Architecture



# Time



.conf2016

splunk >

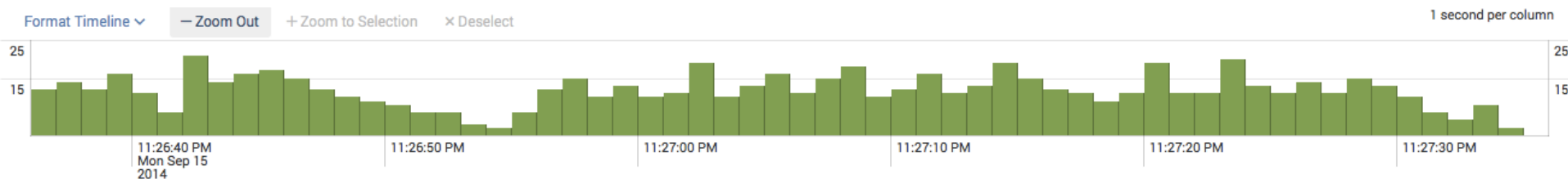
*Be not three hours to my son  
as a minute to you*

*William Shakespeare*



# Time Is Important

- Events are indexed and saved to disk by time
- Searches are bound by a time range
- Events are retrieved based on the parsed time
  - Whatever is in the `_time` field



# The Side-effects

- Incorrectly time stamped events can:
  - Exclude key events from the result set
  - Exclude events from Real Time Window
  - Miss correlation of events
  - Missed alerts
  - Throw off Summary Indexing results
  - Produce incorrect reports
  - Extend your retention windows
  - Create an explosion of buckets
  - Cause heart burn at audit time

# Assigning Time

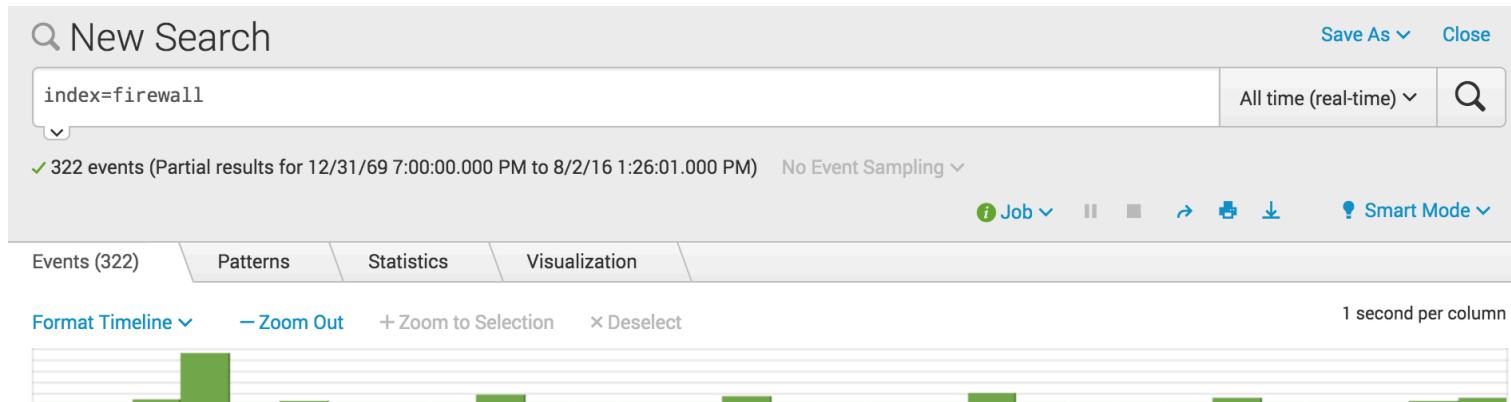
- Stored in the `_time` field in UTC time format
- Determined and set at index time
  - Munge-able at search time using EVAL
    - Requires wacky time ranges to capture events = inefficient
  - Permanent in the Index
    - No-do overs, you must re-index
- Where do we find the time?
  1. The raw event
  2. The local time on the Indexer

# Time Zones

- Splunk converts *\_time* to UTC
- TZ Offset is needed to display *\_time* correctly to the user
- Where do we look for the TZ?
  1. The raw event
  2. TZ parameter in props.conf
  3. The TZ provided by the Universal Forwarder (Splunk 6+)
  4. The TZ of the Indexer

# Finding Time Parsing Issues

- Where are my events? Possibly in the future.
- Run a Real-Time All Time Search
  - One of the only times you should use this



# Other Time Errors

- Check splunkd.log for TimeParserVerbose warnings:

```
index=_internal source=*splunkd.log WARN DateParserVerbose
```

```
08-10-2014 05:59:14.488 +0000 WARN DateParserVerbose - Failed to parse
timestamp. Defaulting to timestamp of previous event. Context="source::/log/
applog.log|host::appserver|appserver_log|remoteport::58689" Text="columnCount
= 6; ..."
```

- These errors are normally due to incorrect TIME\_FORMAT or LINE\_BREAKER errors

# How Do I Fix Parsing Issues?

## Props.conf

```
08-10-2016 05:59:14.488 +0000 WARN CJ029 - Session 18747128 permitted. Connecting to 10.0.0.54
```

```
[source::transaction.log]
```

```
TIME_PREFIX = ^
```

← RegEX that tells Splunk where to start looking for the timestamp

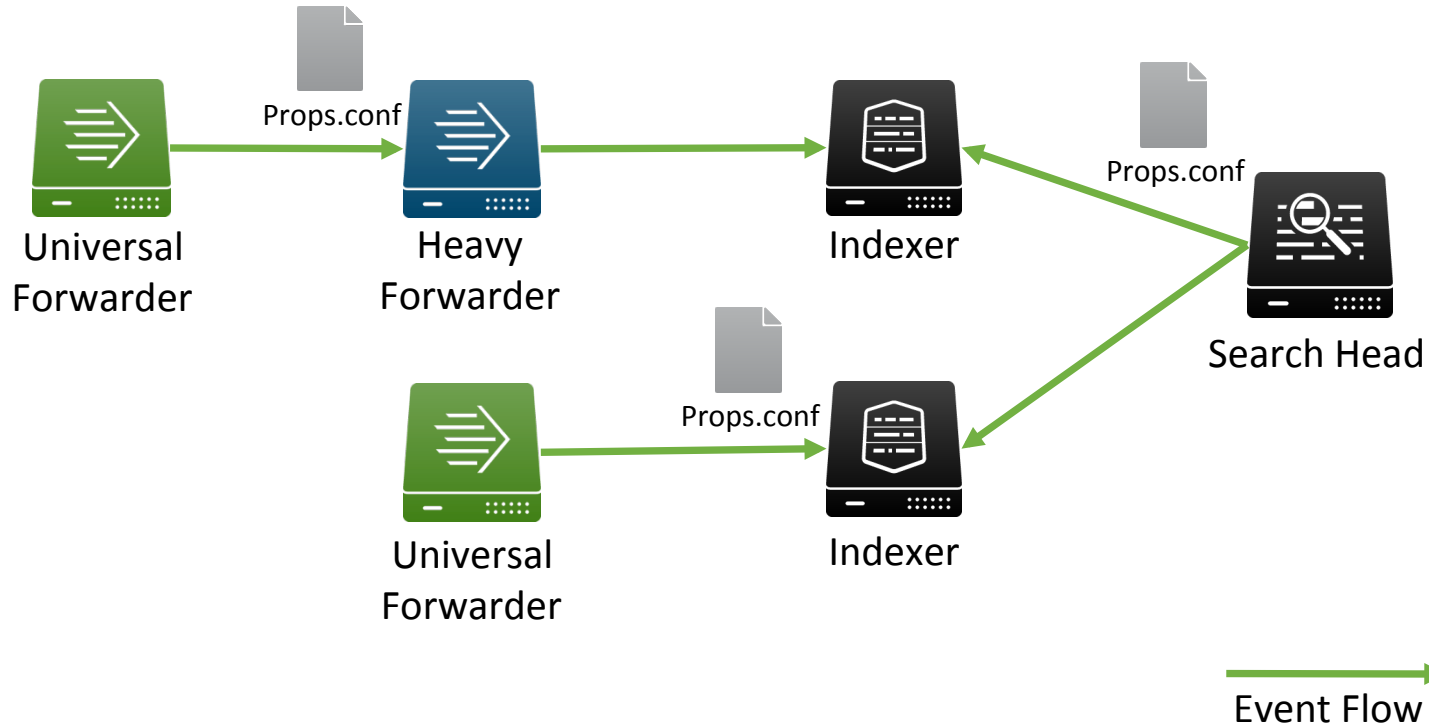
```
TIME_FORMAT = %Y-%m-%d %H:%M:%S
```

← Strptime format that tells Splunk what the timestamp looks like

```
MAX_TIMESTAMP_LOOKAHEAD = 70
```

← # of characters Splunk will read into the log to locate the timestamp

# Where To Apply Props.conf?





# Verify Events Are Matching TIME\_FORMAT

- *08-07-2014 14:33:09.566 -0700*
- Use regex in your search to isolate events which don't match your TIME\_FORMAT

```
index=x sourcetype=y | regex _raw!="^\d{2}-\d{2}-\d{4}\s\d{2}:\d{2}:\d{2}.\d{3}\s-\d{4}"
```

- Similar approach using the PUNCT field
  - Yay, you can use PUNCT for the first time ever!

```
index=x sourcetype=y punct!="__ : : __"
```

# Want To Know More?

**It's 10:00 PM. Do you know where your data is?**

by Jason Timlin

– Thursday, Sept 29<sup>th</sup> 11:20AM – 12:05PM

# Lagging Events (More Like DRAGGING Events...amiright?!



.conf2016

# Causes For Reported Indexing Lag

- Network throughput issues
- Universal Forwarder Throttling (limits.conf)
  - [thruput]  
maxKBps = 256
- Blocked indexer queues
- Buffered writing to log files
- Batch log files or historical log file loads
- Incorrect Timestamp extraction

# Getting Data In On Time

- Events are slow getting into Splunk
- Evaluating your lag
  - *\_time* = The time we parsed from the event
  - *\_indextime* = The time we indexed the event

```
source=<your delayed source> | eval delay=_indextime-_time | fields delay
```



Create a new field called *delay* that is the difference between the index time and the timestamp on the event

# Advanced Lag Monitoring

- Intensive Search
  - Don't run this all the time
  - Taken from S.O.S. – Splunk on Splunk (RIP)

```
index=* OR index=_internal
| eval latency=round((_indextime - _time),2)
| eval seconds_elapsed=(time() - now())
| eval secs=if(seconds_elapsed<0,"1",seconds_elapsed)
| eval esize=(len(_raw)/1024)
| eventstats max(secs) AS seconds
| eventstats count AS ecount, sum(esize) as sum_esize by "sourcetype"
| stats last(ecount) AS "event count", last(eval(ecount/seconds)) AS eps, last(eval(sum_esize/seconds)) AS KBps, min(latency) AS "minimum latency (seconds)", avg(latency) AS
avglat, max(latency) AS "maximum latency (seconds)" min(_time) AS oldestTime max(_time) AS newestTime by "sourcetype"
| eval avglat=round(avglat,2)
| eval eps=round(eps,2)
| eval KBps=round(KBps,2)
| convert timeformat="%m/%d/%Y %H:%M:%S" ctime(newestTime)
| convert timeformat="%m/%d/%Y %H:%M:%S" ctime(oldestTime)
| rename newestTime AS "Time stamp of newest event" oldestTime AS "Time stamp of oldest event" avglat AS "average latency (seconds)" eps AS "events per second" KBps AS
"indexing rate (KBps)"
```

sourcetype	event count	events per second	indexing rate (KBps)	minimum latency (seconds)	average latency (seconds)	maximum latency (seconds)	Time stamp of oldest event	Time stamp of newest event
splunk_web_access	27	8.57	3.68	-0.59	0.58	2.18	08/02/2016 15:25:33	08/02/2016 15:25:52
splunk_web_service	9	2.86	0.40	-0.16	-0.05	0.07	08/02/2016 15:25:38	08/02/2016 15:25:39
splunkd_access	32	10.16	1.53	-0.56	0.71	2.17	08/02/2016 15:25:33	08/02/2016 15:25:52
splunkd_ui_access	77	24.44	25.23	-0.68	0.32	2.19	08/02/2016 15:25:33	08/02/2016 15:25:52

# Indexer Health



.conf2016

# Server Health

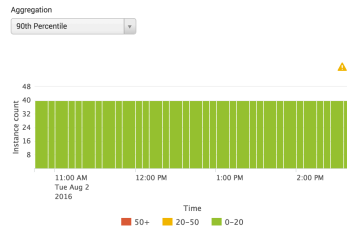
- Key Resources to Monitor
  - Memory Usage
  - CPU load
    - % consumed by Splunk processes
  - I/O throughput
- Commands that should be your friend
  - top
  - free/swap -l
  - iostat
- Splunk Monitoring Console



# Resource Usage: Deployment

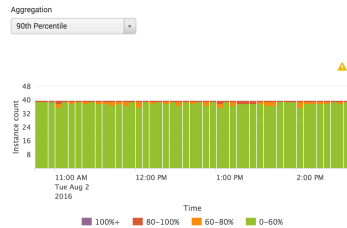
Instance ▾	Load Average ▾	CPU Cores ▾	CPU Usage (%) ▾	Physical Memory Capacity (MB) ▾	Physical Memory Usage (MB) ▾	Physical Memory Usage (%) ▾	I/O Operations per second (Mount Point) ▾	I/O Bandwidth Utilization (Mount Point) ▾
ch-demo-cisco.hod.cloud	1.13	8	94.78	11912	1885	15.83	59 (/four)	3.98% (/four)
ch-demo-pci.hod.cloud	0.40	8	55.79	11912	4615	38.74	185 (/four)	9.95% (/four)
ch-demo-zeus	0.38	8	51.82	11912	3777	31.71	236 (/four)	6.04% (/four)

90th Percentile Load Average



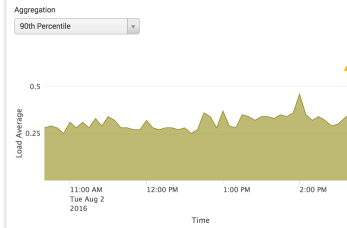
Count of instances grouped by 90th Percentile machine-wide load average over time.

90th Percentile CPU Usage

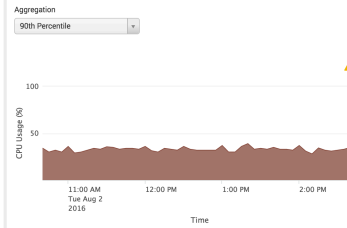


100%+ 80-100% 60-80% 0-60%

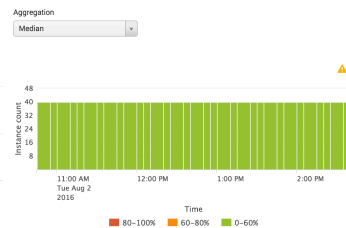
Deployment-Wide 90th Percentile Load Average



Deployment-Wide 90th Percentile CPU Usage

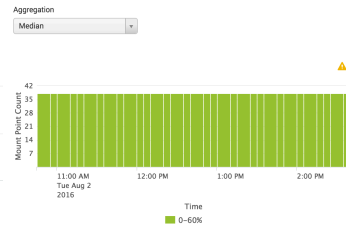


Median Physical Memory Usage



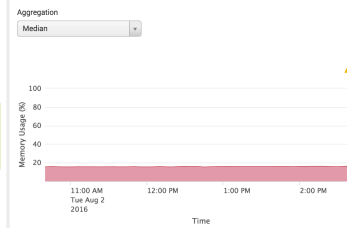
Count of instances grouped by Median machine-wide physical memory usage over time.

Median I/O Bandwidth Utilization

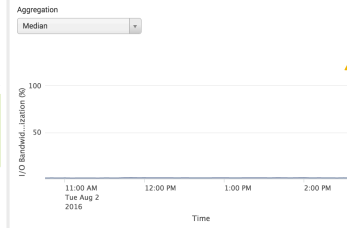


0-60%

Deployment-Wide Median Physical Memory Usage



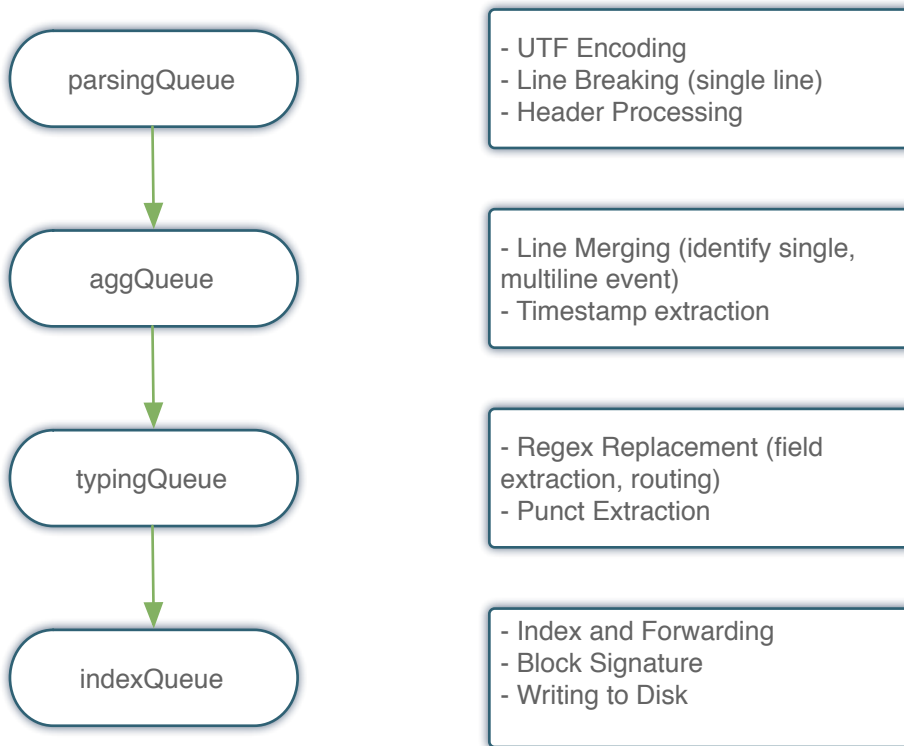
Deployment-wide Median I/O Bandwidth Utilization



# Resource Usage: Deployment

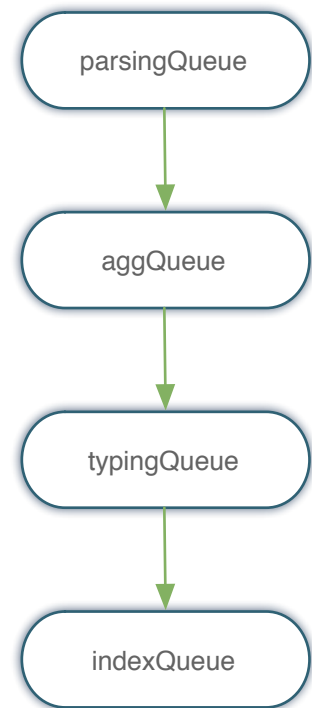
- Look at different levels of aggregation
  - Median, 90<sup>th</sup> Percentile, Maximum
- More detail: Resource Usage: Instance/Machine
- Load Average at or near 1
  - Indicate your system is at capacity
- I/O Bandwidth Utilization
  - Instances must be running Splunk 6.4+
  - Indicates bandwidth utilization not throughput
- I/O Operations Per Second
  - Sum of read & write requests sent per second to the mount point
  - Indicates throughput, but doesn't directly translate to what Splunk considers IOPS (random seeks)

# Important Indexer Queues



# Indexer Queues

- Provide insight into an indexer's internal state
- Blocked queues
  - Indicate a processing bottleneck in the indexing process
  - Will delay the indexing of events
- Default max queue size = 1000
  - Queue will be blocked when limit is reached
- It's normal for queues to be temporarily blocked
  - Indicates that your indexer is doing work
  - Persistent blocked queues indicate a performance issue
    - Will block upstream queues as they fill (backpressure)



# Looking For Bottlenecks

## The Good 'ol Fashioned Way

- View the current queue size by searching the metrics log:

```
index=_internal source=*metrics.log group=queue | timechart median(current_size) by name
```

_time	OTHER	aeq	aggqueue	aq	auditqueue	exec	indexqueue	parsingqueue	tcpout_default-autolb-group	tcpout_sosdev-idx	typingqueue
2016-08-03 09:30:30	0	0	0	0	0	0	0	0	18356	200	0
2016-08-03 09:30:40	0	0	0	0	0	0	0	0	1004	600	0
2016-08-03 09:30:50	0	0	238	0	0	0	0	0	5087	9336	143

- Find blocked queue events

```
index=_internal source=*metrics.log group=queue blocked
```

Time	Event
8/3/16 9:45:00.063 AM	08-03-2016 09:45:00.063 -0700 INFO Metrics - group=queue, ingest_pipe=1, name=rsingqueue, blocked=true, max_size_kb=6144, current_size_kb=6143, current_size=1475, largest_size=1486, smallest_size=1246 group = queue   host = svdev-centos6-002.sv.splunk.com
8/3/16 9:44:25.413 AM	08-03-2016 09:44:25.413 -0700 INFO Metrics - group=queue, name=aggqueue, blocked=true, max_size_kb=1024, current_size_kb=1023, current_size=1067, largest_size=3004, smallest_size=142 group = queue   host = svdev-fedora14-05

# Looking For Bottlenecks

## Splunk Monitoring Console

- Indexing Performance: Instance

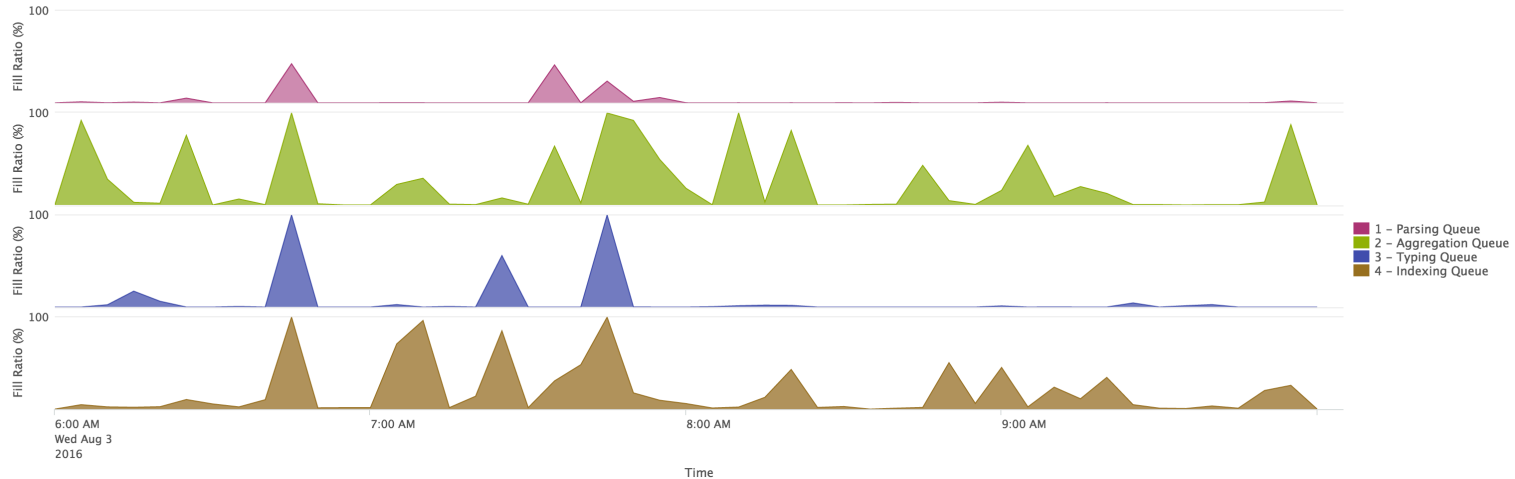
90th Percentile Fill Ratio of Data Processing Queues

Queues to Measure:

Event-Processing Queues

Aggregation

90th Percentile

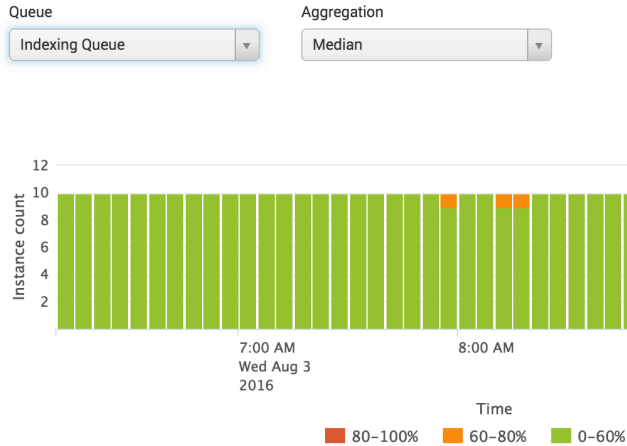


# Looking For Bottlenecks

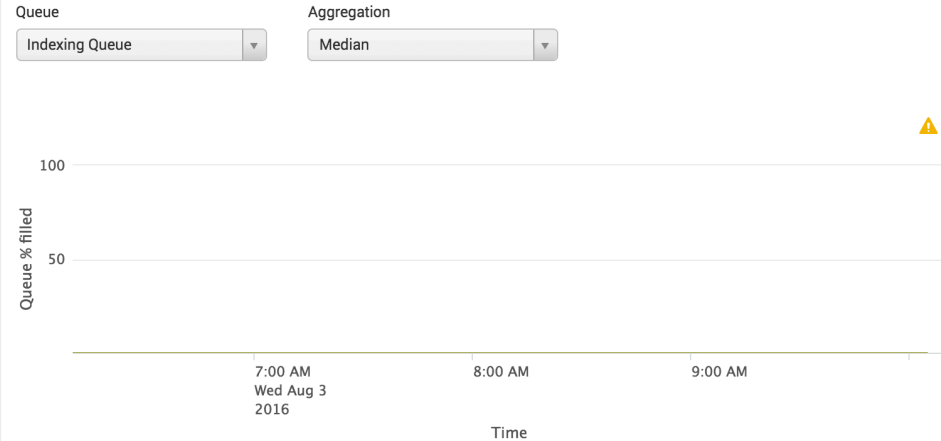
## Splunk Monitoring Console

- Indexing Performance: Deployment

Instances By Median Queue Fill Ratio

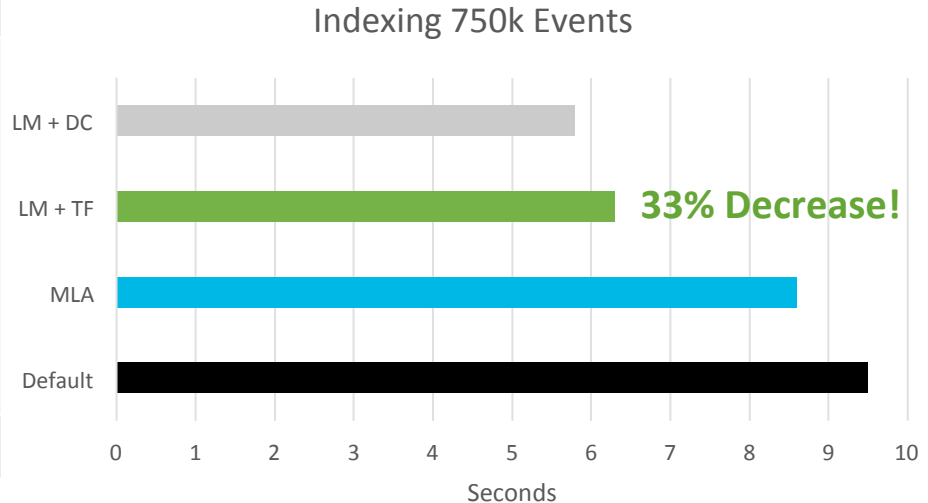


Deployment-Wide Median Queue Fill Ratio



# Tuning Configs

Conf File	props.conf
Parameter	<code>line_breaker =</code> (LM) <code>should_linemerge = false</code> <code>time_prefix =</code> (MLA) <code>max_timestamp_lookahead =</code> (TF) <code>time_format =</code> <code>truncate =</code> <code>tz=</code> (DC) <code>datetime_config = current</code>
Splunk Version	5.x, 6.x



- Time spent in `parsingQueue` & `aggQueue` will SIGNIFICALLY improve
  - `line_breaker=` goes hand-in-hand with `should_linemerge=`
  - `[datetime_config = current]` does not extract a timestamp, it sets `_time` to the local indexer time



# Tuning Configs

## typingQueue

- props.conf
  - TRANSFORMS-xxx
  - SEDCMD
- transforms.conf
  - SOURCE\_KEY
  - DEST\_KEY
  - REGEX
  - FORMAT

## indexerQueue

- Index and Forwarding
- Block Signing
- indexes.conf settings
  - maxHotBuckets = 10
  - maxDataSize = auto\_high\_volume

- In most cases a blocked indexerQueue is due to more data being written to disk than can be serviced
- If the disk performance cannot be improved, consider adding additional indexers to offload this indexing workload

# Search Performance

.conf2016

splunk >

# Tackling Search Performance

- Identify slow/resource intensive searches
  - Syntax efficiencies
  - Better utilization of MapReduce
- Summary Indexing & Search Accelerations
  - Considerations
  - Using time snaps
- Scheduler Efficiencies
  - Avoiding hot spots at the top/bottom of the hour
  - Scheduler Windows
- Bundle Replication White/Blacklists

# Identifying Slow Searches

## Splunk Monitoring Console

### Search > Activity > Search Usage Statistics: Deployment

- Search Activity by User

User	Search Count	Search Head Count	Median Runtime	Cumulative Runtime	Last Search
admin	602	2	0.9 sec	533.0 sec	08/03/2016 12:16:15 -0700
splunk-system-user	24	1	0.2 sec	4.7 sec	08/03/2016 12:12:35 -0700

- Who is running the most searches?
- Which are taking the longest?

- Frequently Run Searches

Report Name/Search String	Search Runtime	Search Start	Earliest Time	Latest Time	Type	User	Host	SID
metadata type=sourcetypes   search totalCount > 0	7.8 sec	08/03/2016 09:45:24 -0700	all time	all time	ad hoc	admin	dmc- demo.sv.splunk.com	rt_1470242724.57958
search index=_internal earliest=-1d   head 30000	2.9 sec	08/03/2016 10:15:01 -0700	all time	all time	ad hoc	admin	sosdev-sh1	1470244501.945967

- Which searches are run often?
- Can they be improved?
- Can we move it to a scheduled search?

- Long Running Searches

Report Name/Search String	Count	Median Runtime	Max Runtime	Users	Hosts	Type
search index=_internal earliest=-1d   head 30000	475	0.9 sec	2.9 sec	admin	sosdev-sh1	ad hoc
rest /services/licenser/slaves   rename label AS indexer_name title AS indexer_guid   table indexer_guid indexer_name	24	0.2 sec	0.3 sec	splunk-system-user	sosdev-sh1	ad hoc

- Who are my worst offenders?
- Can they be improved?

# Identifying Slow Searches

## Splunk Monitoring Console

### Search > Activity > Search Activity: Deployment

- Top 20 Memory Consuming Searches

	Name ↕	Memory Usage (MB) ↕	Instance ↕	Runtime ↕	Started ↕	Type ↕
1	Latex R&D report - Daily	2010.54	sosdev-cm1	577.9 sec	Wed Aug 3 09:58:09 PDT 2016	scheduled
2	historical scheduled search - m30	329.61	svdev-sh-demo	468.3 sec	Wed Aug 3 12:10:09 PDT 2016	scheduled
3	DM indexthru week over week	154.06	sosdev-sh1	2.0 sec	Wed Aug 3 09:30:03 PDT 2016	scheduled
4	1470248762.946285	115.54	sosdev-sh1	1.0 sec	Wed Aug 3 11:26:03 PDT 2016	ad-hoc

- Which searches are impacting system resources?
- Better ways to filter results?
- More efficient commands?
- Re-order commands?

### Search > Activity > Search Usage Statistics: Instance

- Common Search Commands

	Command ↕	Count ↕	Average Runtime ↕	Max Runtime ↕
1	search	5	2.3 sec	7.8 sec
2	metadata	1	7.8 sec	7.8 sec
3	timechart	1	0.5 sec	0.5 sec

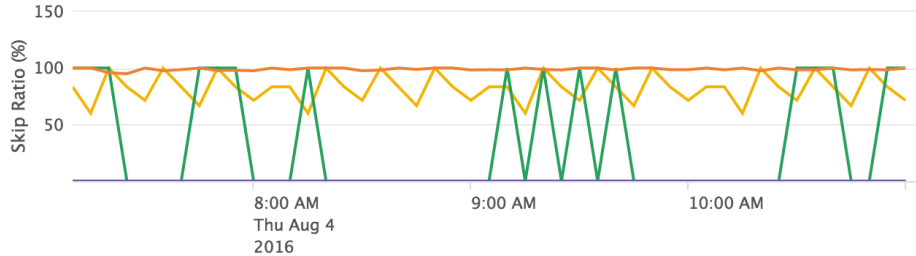
- Alternatives for slow commands?
  - Join/Transaction → stats
- Efficient use of MapReduce?
  - Stats → prestats
  - sort → presort

# Skipped Searches

## Splunk Monitoring Console

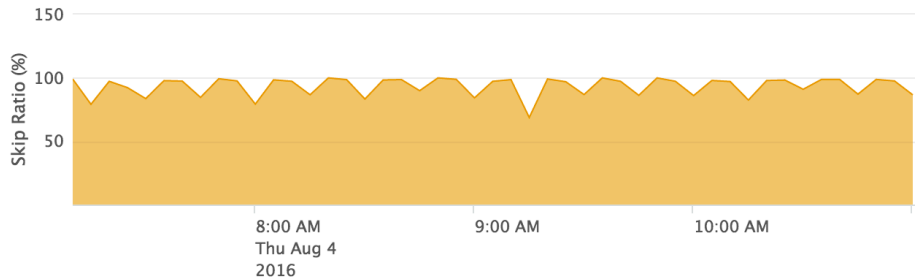
### Search > Scheduler Activity: Deployment

- Skip Ratio by Instance



- Where are jobs being skipped?
- When are they being skipped?
- Do I have enough capacity?
- Can I adjust schedules?

- Skip Ratio Across All Instances



- What % of all jobs are being skipped across my deployment?

# Skipped Searches

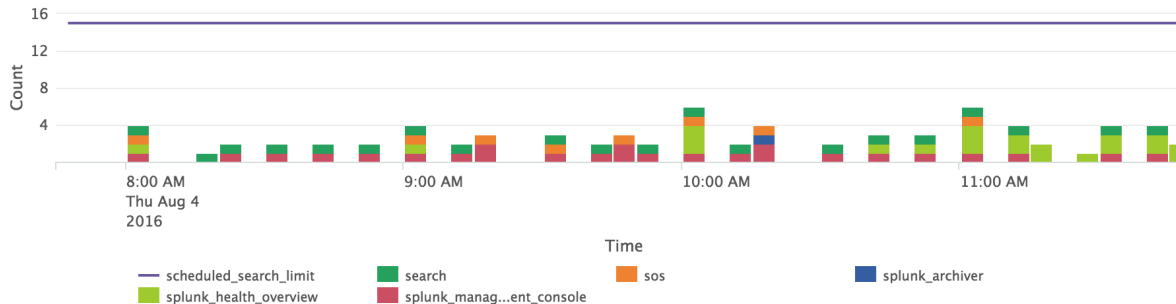
## Splunk Monitoring Console

### Search > Scheduler Activity: Instance

- Runtime Statistics

Report Name	App	User	Cron Schedule	Schedule Interval (sec)	Average Runtime (sec)	Interval Load Factor	Total Executions	Skipped Executions	Skip Ratio
<a href="#">sos_refresh_splunk_forwarders_cache</a>	sos	nobody	3,18,33,48 ****	900	1	0.11 %	12	0	0.00 %
<a href="#">DMC Forwarder - Build Asset Table</a>	splunk_management_console	nobody	3,18,33,48 ****	900	1	0.11 %	12	0	0.00 %

- Median Concurrency of Scheduled Reports



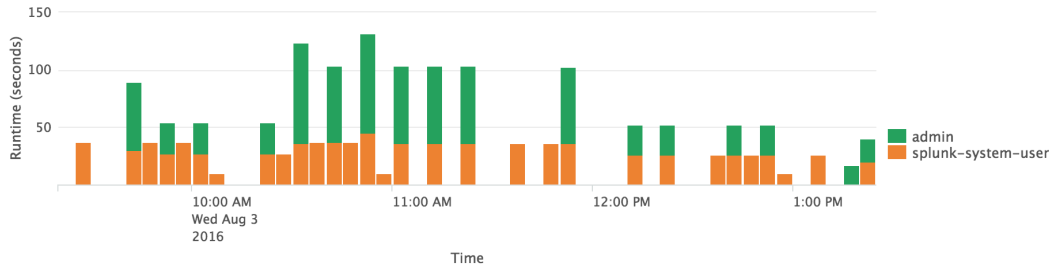
- Which jobs are getting skipped and how often?
- How long are these jobs running?
- Which apps or users are generating the most searches?

# Search Runtime

## Splunk Monitoring Console

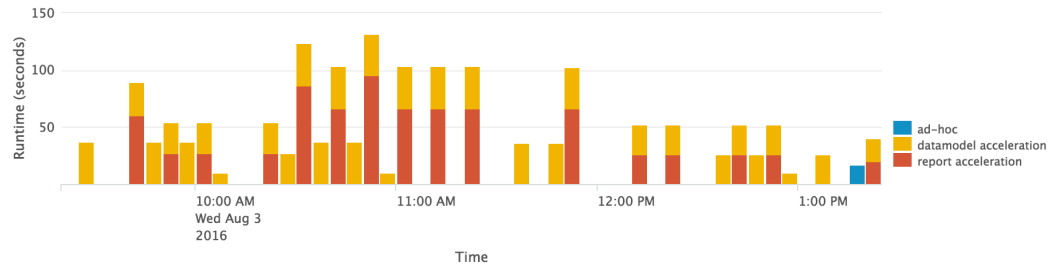
### Search > Activity > Search Activity: Instance

- Aggregate Search Runtime by User



- Who is consuming search capacity?
- When are jobs being run?
  - Can we avoid peak times?

- Aggregate Search Runtime by Type



- What types are searches are consuming capacity?
- Can I move anything to search windows?

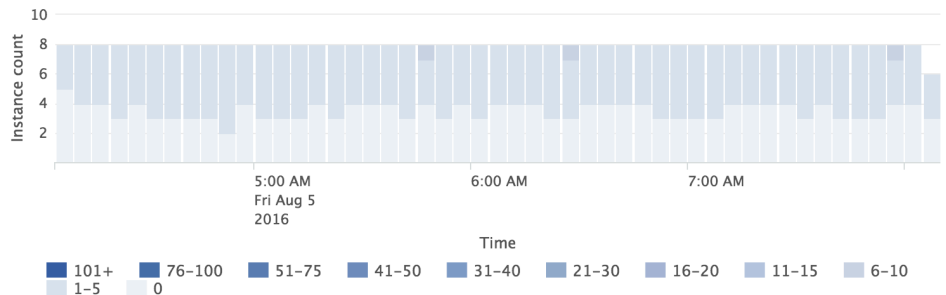


# Search Concurrency

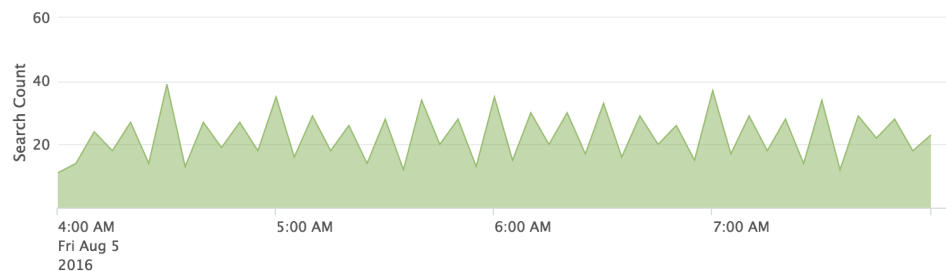
## Splunk Monitoring Console

### Search > Activity > Search Activity: Deployment

- Instances by 90<sup>th</sup> Percentile Search Concurrency



- Deployment-Wide Total Concurrent Searches



- Helpful to look at 90<sup>th</sup> Percentile and Median Concurrency
- Look at SH & Indexers
- Instances are grouped by concurrency
- Count of the unique searches running concurrently over time

# Concurrent Search Limit

- Maximum # of Concurrent Searches per SH Instance
  - $(max\_searches\_per\_cpu \times \text{Logical \# of CPUs}) + base\_max\_searches$

Conf File	limits.conf
Parameter	[search] base_max_searches = 6 max_searches_per_cpu = 1
Splunk Version	5.x, 6.x

- Logical # of CPUs as reported by OS
  - Hyper-threading enabled = 2 x physical cores
- Use caution before raising these settings
  - May increase the CPU load and the time taken for any search to complete!
  - Search Heads queue, indexers do not
    - Don't crash your indexers

# Scheduled Search Limit

- The ratio of jobs that the scheduler can use vs. ad-hoc searches

<b>Conf File</b>	limits.conf
<b>Parameter</b>	[scheduler] max_searches_perc = 25* max_searches_perc = 50
<b>Splunk Version</b>	*5.x, 6.x

- Default changed from 25% to 50% in Splunk 6.0+
  - In 6.0+, you can have different settings depending on the time/day
- If you have a large number of Scheduled Searches increasing this percentage may help with skipped searches
- The searches are NOT reserved for the scheduler - this is just a limit
  - Ad-hoc user searches take priority over the scheduler

# Scheduler Windows

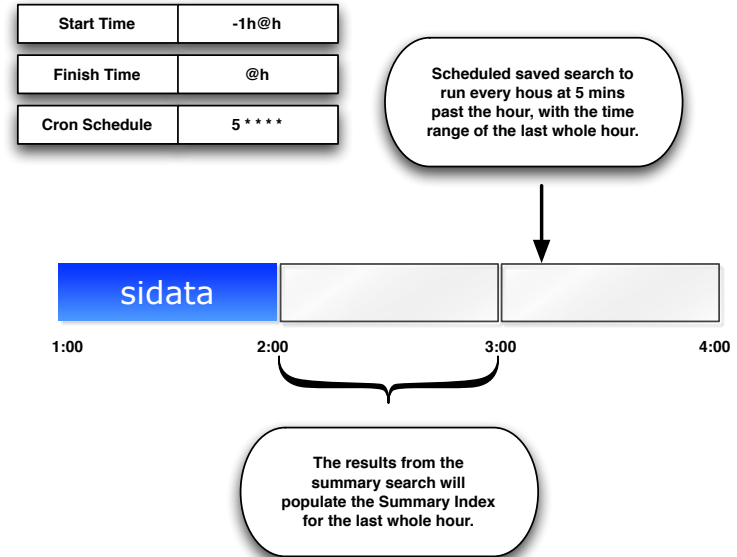
- Good for jobs that don't have to run at a specific time
  - Ex: Run this search anytime between midnight and 5am
- Search Windows help other searches
  - Free up scheduler time for searches that must run at a specific time
  - Helps Splunk determine long the search can be deferred
- Should not be used for searches that run every minute
- Search Window must be less than the search frequency
  - Ex: If my search needs to run every hour, but I set a 2 hour window, my search may be missed

# Accelerations & Summaries

- Data Model Accelerations
  - Speed up reporting against the entire set of fields defined in the model
  - Fields/attributes in the model are stored in a special TSIDX (high-performance analytics store)
  - Backfill is handled automatically
- Report Accelerations
  - Speed up individual searches/reports
  - Pre-computed summaries from the report are stored in a special TSIDX
  - Backfill is handled automatically
- Summary Indexes
  - Speed up individual searches/reports
  - Pre-computed summaries or a subset of data is stored in a new index
  - Backfill must be handled manually
  - Allows for retention periods that are different from the source index

# Summary Indexing

- Use time modifier offsets to snap-to-time
- For last hour “-1h@h to @h”
- Offset the scheduled time from the top of the hour to 5 mins after the hour “5 \* \* \* \*”
- This offset method can be applied to other scheduled searches



# Bundle Replication

## What are Knowledge Bundles?

- Replicated from Search Head to all Indexers every minute
  - Replication only happens if changes occur
  - Splunk attempts to replicate deltas when possible
- Indexers must have a common KB before a search can continue
- Large KBs can cause search execution to be delayed
  - Typically caused by very large lookup files that are updated frequently
  - Monitoring Console > Search > Distributed Search: Deployment
- You can limit the size of KBs by using whitelists/blacklists
  - Must understand where the knowledge object will be used before modifying
  - Ex: *Is the lookup going to be used on the Indexers or on the Search Head?*
    - This depends on how you've structured your search commands

# But Wait, There's More...




















.conf2016

splunk >



# Monitoring Console Health Check

16	1	4	1	11	0
All	Error	Warning	Info	Success	N/A
Check ↕				Category ↕	Results ↕
Missing Forwarders				Data Indexing	 1 (100%)
System hardware provisioning assessment				System and Environment	 6 (100%)
Assessment of server ulimits				System and Environment	 6 (100%)
Linux kernel transparent huge pages				System and Environment	 6 (100%)
Expiring or expired licenses				Data Indexing	 1 (100%)
Event-processing issues				Data Collection	 2 (100%)
Local indexing on non-indexer instances				Data Indexing	 4 (100%)
Distributed search health assessment				Data Search	 4 (100%)
Search Scheduler Skip Ratio				Data Search	 4 (100%)
Indexing status				Data Indexing	 5 (100%)
Near-critical disk usage				System	 1 (16%)  5 (83%)
Saturation of event-processing queues				Data Indexing	 6 (100%)
Integrity check of installed files				Splunk Miscellaneous	 6 (100%)
Orphaned scheduled searches				Splunk Miscellaneous	 6 (100%)
Excessive Physical Memory Usage				Splunk Miscellaneous	 6 (100%)
License warnings and violations				Data Indexing	 6 (100%)

- New in Splunk 6.5
- Checks for common issues
- Focused on Healthy/Unhealthy
  - When things are bad, we provide specific instructions for resolving the issue

# Running Linux?

## Turn Off Transparent Huge Pages

- 30% performance impact on search and indexing

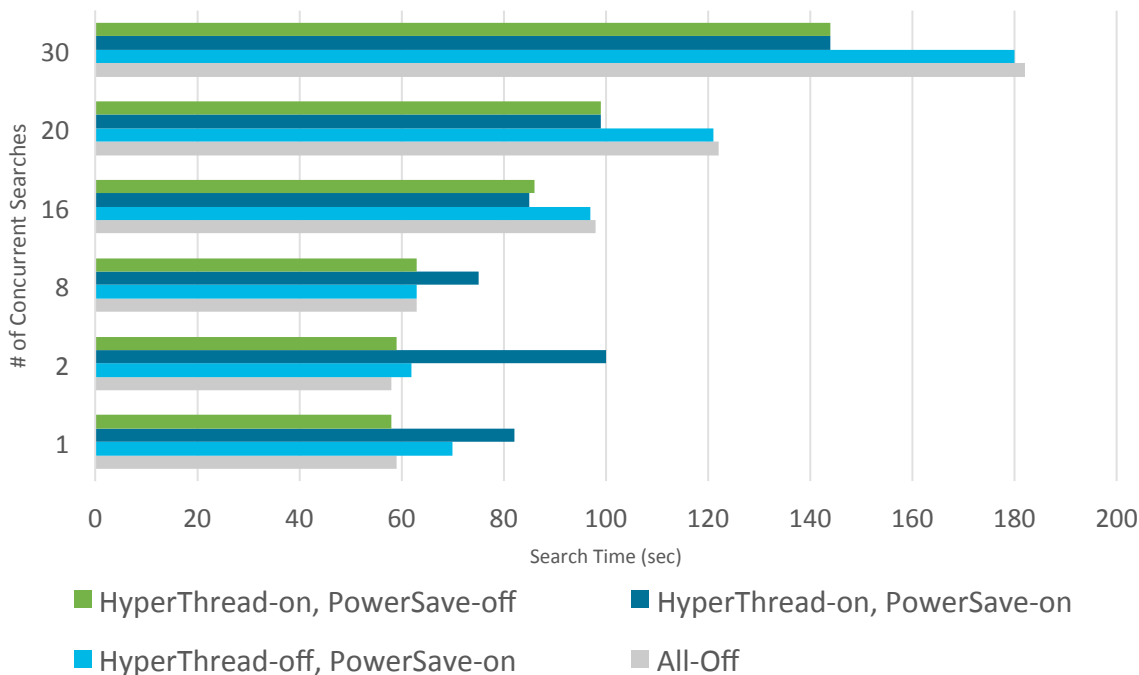
OS	Linux
Paths	/sys/kernel/mm/transparent_hugepage/ enabled  /sys/kernel/mm/transparent_hugepage/defrag
Splunk Version	5.x, 6.x

- Confirm the path for your specific OS
  - Ensure your method for disabling THP survives a reboot
  - <https://answers.splunk.com/answers/188875/how-do-i-disable-transparent-huge-pages-thp-and-co.html>
- Splunk will log THP status on startup in splunkd.log
  - Regularly check to ensure this is still off
  - SAMPLE: 11-18-2014 08:19:42.052 -0600 INFO ulimit - Linux transparent hugetables support, enabled="never" defrag="never"

# Running Intel Processors?

## Enable Hyper-threading and Disable Power Saving in BIOS

- Power saving affects low-concurrency workloads most
- HT and power saving can result in a 30% performance impact for high-concurrency workloads
- HT won't hurt you
  - SO TURN THAT \$h\*t on!
- Test Details
  - 12-core Intel Xeon X5690 @ 3.33GHz
  - CPU-bound search jobs over the same sample data
  - Individual search could complete ~60s
  - <https://answers.splunk.com/answers/7845/splunk-and-hyper-threading-has-anyone-run-benchmarks.html>



# Where Can I Learn More?

.conf2016

splunk >

# What Now?

## Related breakout sessions and activities...

- **Worst Practices...and how to fix them** by Jeff Champagne
  - Tuesday, Sept 27<sup>th</sup> 10:30AM – 11:15AM
- **Best and Better Practices for Admins** by Burch Simon
  - Tuesday, Sept 27<sup>th</sup> 11:35AM – 12:20 PM
- **Architecting Splunk for High Availability and Disaster Recovery** by Dritan Bitincka
  - Tuesday, Sept 27<sup>th</sup> 5:25PM – 6:10PM
- **Observations and Recommendations on Splunk Performance** by Dritan Bitincka
  - Wednesday, Sept 28<sup>th</sup> 12:05PM – 12:50PM
- **Behind the Magnifying Glass: How Search Works** by Jeff Champagne
  - Wednesday, Sept 28<sup>th</sup> 1:10PM – 1:55PM
- **Search: Under the Hood** by Chris Pride
  - Wednesday, Sept 28<sup>th</sup> 4:35PM – 5:20PM
- **It's 10:00 PM. Do you know where your data is?** by Jason Timlin
  - Thursday, Sept 29<sup>th</sup> 11:20AM – 12:05PM
- **Making the Most of the Splunk Scheduler** by Paul Lucas
  - Thursday, Sept 29<sup>th</sup> 12:25PM – 1:10 PM
- **Best and Better Practices for Users** by Burch Simon
  - Thursday, Sept 29<sup>th</sup> 12:25PM – 1:10PM

Time Issues  
Search  
Admin

# Resources

- Reporting Manual – Scheduler Windows
  - <http://docs.splunk.com/Documentation/Splunk/latest/Report/Schedulereports>
- Making the Most of the New Splunk Scheduler (.conf 2015)
  - [http://conf.splunk.com/session/2015/conf2015\\_PLucas\\_Splunk\\_SplunkEntWhatsNew\\_MakingTheMostOf.pdf](http://conf.splunk.com/session/2015/conf2015_PLucas_Splunk_SplunkEntWhatsNew_MakingTheMostOf.pdf)
- Splunk Data Pipeline
  - <http://docs.splunk.com/Documentation/Splunk/latest/Deploy/Datapipeline>
- Props.conf Spec File
  - <http://docs.splunk.com/Documentation/Splunk/latest/Admin/Propsconf>
- Limits.conf Spec File
  - <http://docs.splunk.com/Documentation/Splunk/latest/Admin/Limitsconf>
- Overview of summary-based search and pivot acceleration
  - <http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Aboutsummaryindexing>
- Modify the Knowledge Bundle
  - <http://docs.splunk.com/Documentation/Splunk/latest/DistSearch/Limittheknowledgebundlesize>

# Questions?



.conf2016

# THANK YOU

.conf2016