.conf2015

Making the Most of the New Splunk Scheduler

Paul J. Lucas

Sr. Software Engineer, Splunk

splunk>

Disclaimer

Personal Introduction

- Paul J. Lucas, Sr. Software Engineer, Splunk
- On the Core Server Engineering Team
- Worked on Search Scheduler improvements to Splunk Enterprise
- Also worked on various parts of the Deployment Server
- Has been using C++ since the “cfront” days at AT&T Bell Labs

Agenda

- **Scheduled Searches**:
  A. Introduction
  B. How Cron Works
  C. Cron vs. Splunk Scheduler
- **Splunk Scheduler**:
  A. Deferred vs. Skipped
  B. Lag
  C. `limits.conf` Settings
- **Splunk Scheduler Improvements**:
  A. How the Splunk Scheduler Works
  B. Priority Scoring
  C. Schedule Windows
  D. Variable `max_searches_perc`
  E. Continuous Saved Searches “Catch-up”
- **Takeaways**
- **Splunk Enterprise 6.3 Test Results**

Scheduled Searches

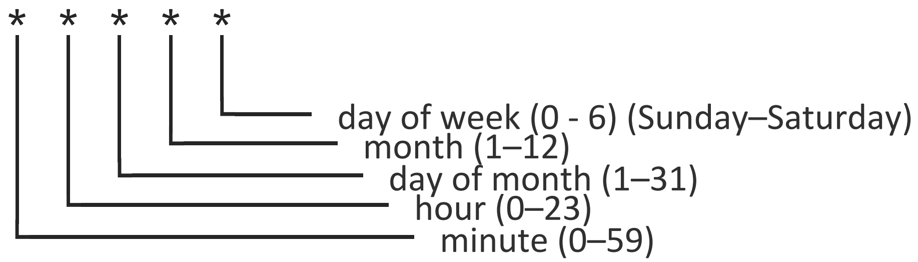
Scheduled Searches: Introduction

- Splunk allows you to save your searches and run them on a schedule
- Scheduled searches can be used to trigger an alert action (possibly when a condition is met) or to speed-up dashboards
- An alert action is either sending an e-mail or running a script
- **Example:** index=_internal source=*splunkd.log* error NOT debug

Title	Too many errors
Trigger condition	Number of Results
Number of results is	Greater than: 5
in	1 Minute

Scheduled Searches: Introduction

- Scheduling is specified via a five-field cron string:



- All (*), ranges (1–5), lists (1,8,15,22), and "every *n*" (*/6) allowed
- **Example**: `0 */6 1,15 * *` = every 6 hours on the hour on the 1st and 15th

How Cron Works

1. For each cron entry, calculate the next run-time of the command
2. Place all commands in a priority queue by time
3. Enter main loop:
    - A. Examine the entry at the head of the queue
    - B. Calculate the delta between that entry's next run-time and *now*
    - C. If delta > 0, sleep for that period of time
    - D. Run the entry's command (in the background)
    - E. Calculate the *next* run-time of the command and place it back on the queue with that new time value

Cron vs. Splunk Scheduler

Cron

- No job quotas
- Entirely manual scheduling — have to skew searches by hand:

```
0  0 * * * command-1
15 0 * * * command-2
30 0 * * * command-3
45 0 * * * command-4
```

- Limited to a single machine

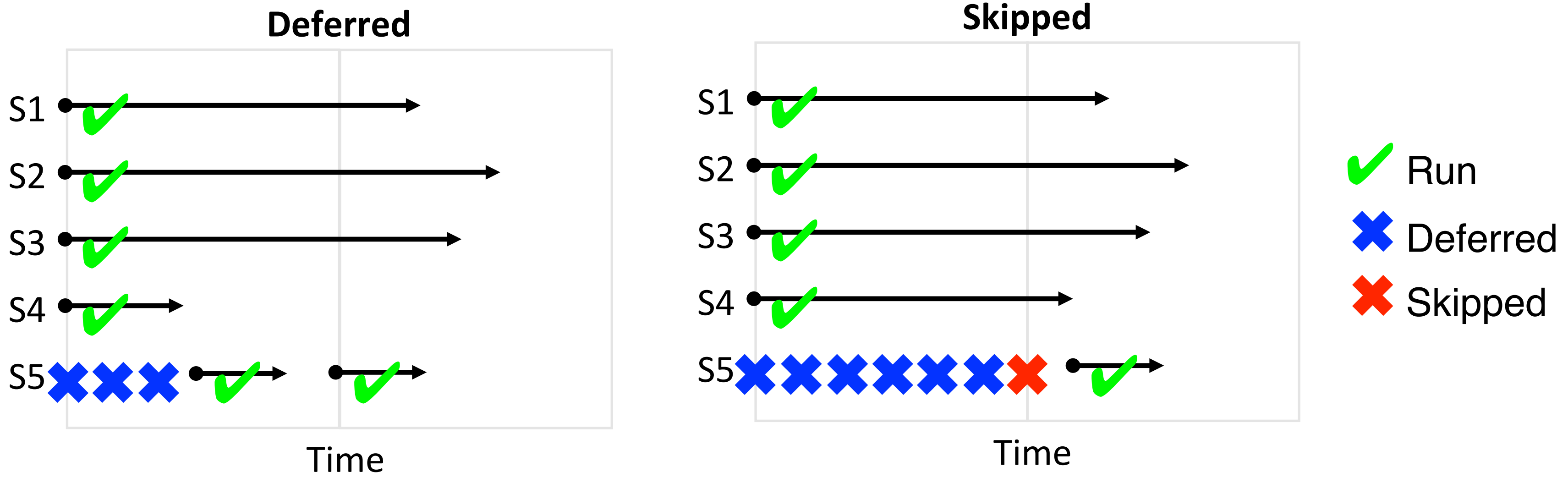
Splunk Scheduler

- Quotas: limit search concurrency — reserves CPU for other tasks
- Searches over quota are *deferred,* but implicitly retried repeatedly for the duration of their periods until either run or *skipped*
- Can distribute searches across a cluster of machines

Splunk Scheduler

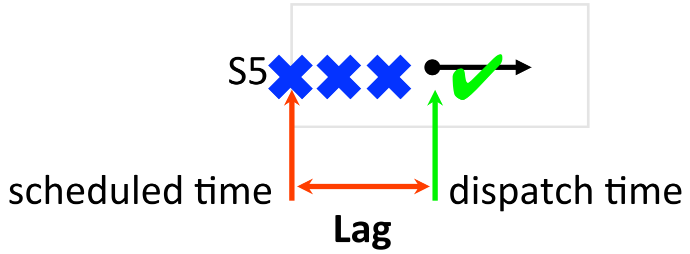
Deferred vs. Skipped

- As mentioned, searches over quota are *deferred*, but are implicitly retried repeatedly for the duration of their periods until either run or *skipped*



Lag

- "Lag" is the difference between a search's *dispatch* time and its *scheduled time*



- Non-zero lag means scheduler is over-subscribed (at least temporarily)
- Causes delays and may cause skipping
- May be mitigated by *schedule windows* (new in 6.3 — more later)

`limits.conf` Settings

- **base_max_searches**: A constant added to max. total searches (default = 6)
- **max_searches_per_cpu**: Maximum number of concurrent searches per CPU (default = 1)
- Given those, the total maximum number of concurrent searches allowed is:

max. total searches = max_searches_per_cpu × *number of CPUs*
+ base_max_searches

- **max_searches_perc**: Maximum number of concurrent searches the scheduler can run as a percentage of max. total searches (default = 50)

limits.conf Settings

- Some example numbers:

CPUs	max_searches_per_cpu	base_max_searches	max. total searches	max_searches_perc	max. scheduled searches
1	1	6	7	50%	3
8	1	6	14	50%	7
64	1	6	70	50%	35

🟨	.conf setting
🟩	Calculated

.conf2015

Scheduler Improvements In Splunk Enterprise 6.3

splunk>

How the Splunk Scheduler Works

1. For each search, calculate the next run-time of the search
2. Place all searches in a map<search_id,next_runtime>
3. Enter main loop:
   A. For each search, if its next run-time ≤ *now*, add it to the candidate search list
   B. Randomly shuffle the candidate list
   C. For each candidate search, calculate its *priority score*
   D. Sort all candidate searches by priority score
   E. For each candidate search, if it doesn't exceed quota, run it, calculate the *next* run-time of the search, and update the map

Priority Scoring

- **Problem in 6.2**: Simple single-term priority scoring could result in saved search lag, skipping, and starvation (under CPU constraint)
- **Solution in 6.3**: Better multi-term priority scoring mitigates problems and improves performance by 25%

$$\begin{aligned} \mathit{score}(j) &= \mathit{next\_runtime}(j) \\ &+ \mathit{average\_runtime}(j) \times \mathrm{priority\_runtime\_factor} \\ &- \mathit{skipped\_count}(j) \times \mathit{period}(j) \times \mathrm{priority\_skipped\_factor} \\ &+ \mathit{schedule\_window\_adjustment}(j) \end{aligned}$$

Schedule Windows

- **Problem in 6.2**: Scheduler can not distinguish between searches that (A) *really should* run at a specific time (just like cron) from those that (B) don't have to. This can cause lag or skipping
- **Solution in 6.3**: Give a *schedule window* to searches that don't have to run at specific times
  Example: For a given search, it's OK if it starts running sometime between midnight and 6am, but you don't really care when specifically
- A search with a window helps *other* searches
- Search windows *should not* be used for searches that run every minute
- Search windows *must* be less than a search's period

Variable **max_searches_perc**

- **max_searches_perc**: Maximum number of concurrent searches the scheduler can run as a percentage of max. total searches (default = 50)
- **Problem in 6.2**: Constant value limits scheduler during off-peak times
- **Solution in 6.3**: Allow **max_searches_perc** to vary by time or day:

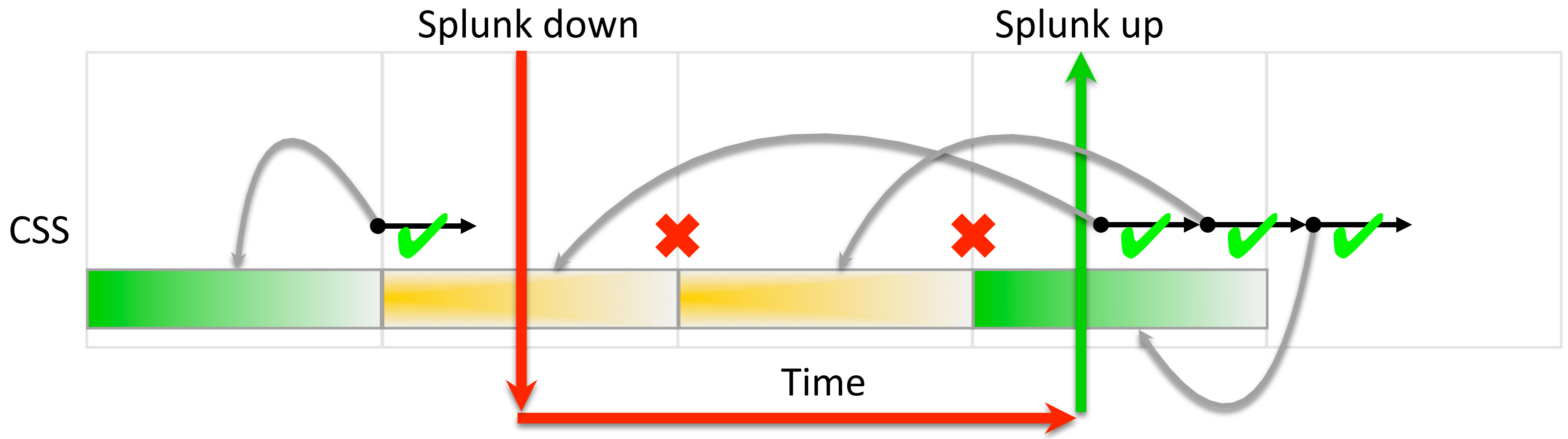
```
max_searches_perc = 50

# Allow value to be 75 anytime on weekends.
max_searches_perc.1 = 75
max_searches_perc.1.when = * * * * 0,6

# Allow value to be 90 between midnight and 5am.
max_searches_perc.2 = 90
max_searches_perc.2.when = * 0-5 * * *
```

Continuous Scheduled Searches “Catch-up”

- **Problem in 6.2**: Continuous Scheduled Searches (CSSs) are missed due to Splunk downtime creating data gaps
- **Solution in 6.3**: By remembering last execution time, missed CSSs are run as soon as Splunk comes back up to fill in data gaps



Splunk Enterprise 6.3 Test Results

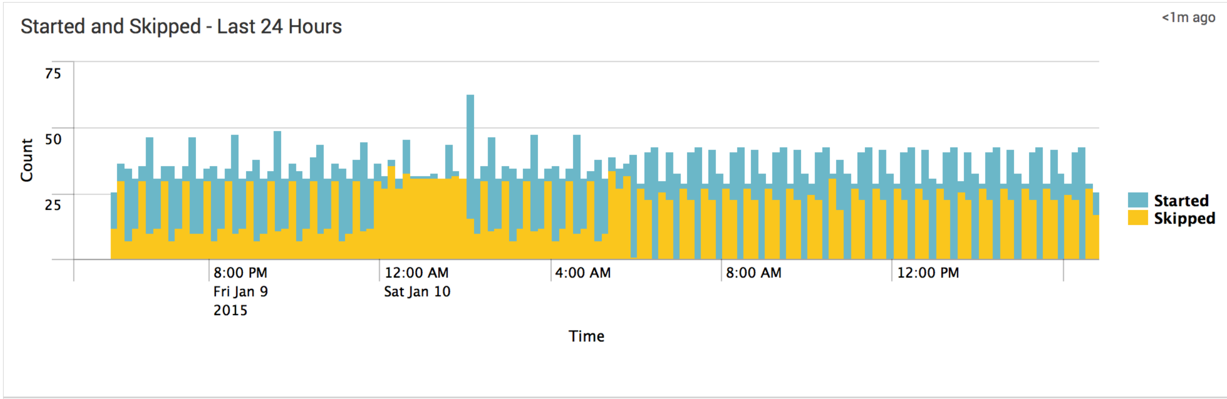
Testing Conditions

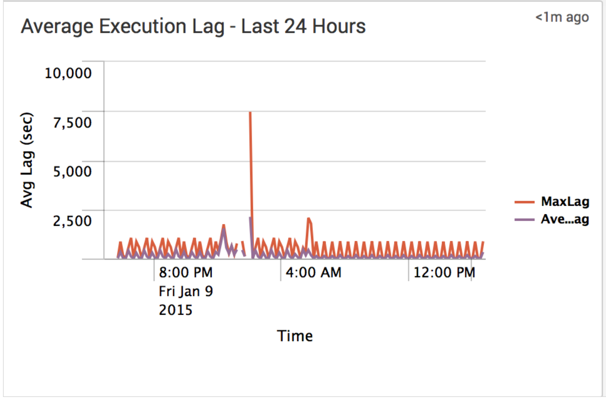
- Stand-alone instance of Splunk (non-cluster)
- Given lots of searches and intentionally hamstrung by a low (15%) `max_searches_perc` to test performance under duress
- In production, for the same number of searches, you should have a higher `max_searches_perc`, a bigger CPU, or a search cluster

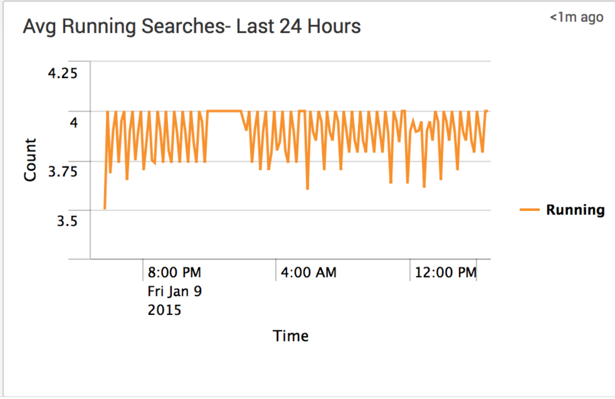
Splunk 6.2 Test Results

Things to notice:

- Started/skipped forms pattern: same searches are run in same order
- From 12–2am, many searches are skipped
- Average running searches is erratic



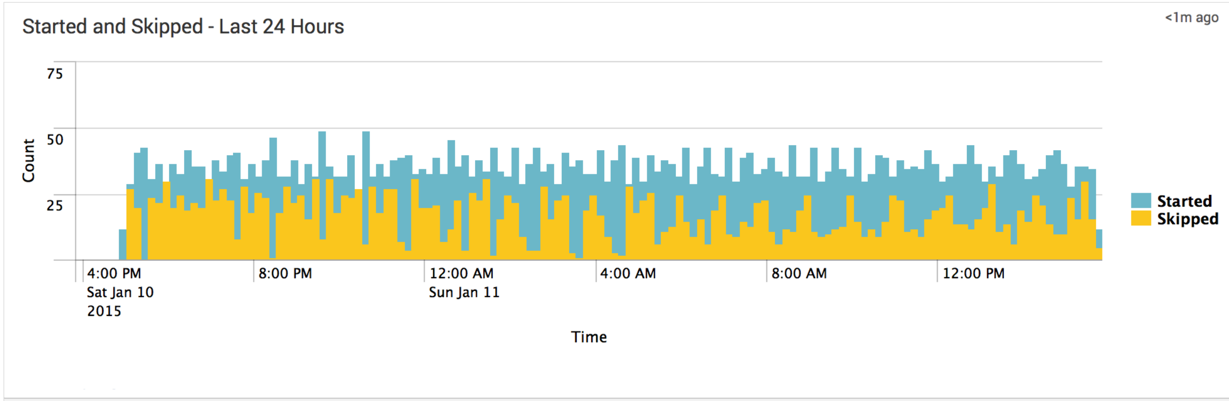


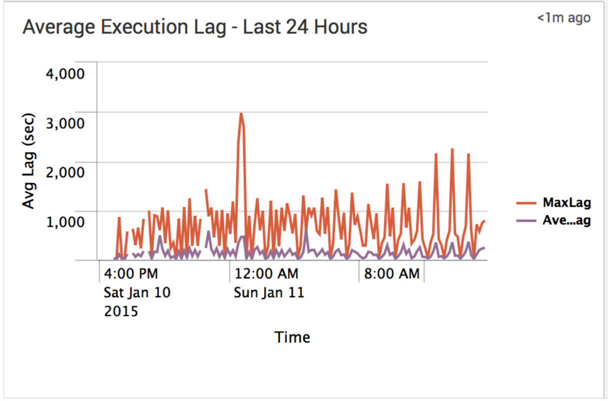


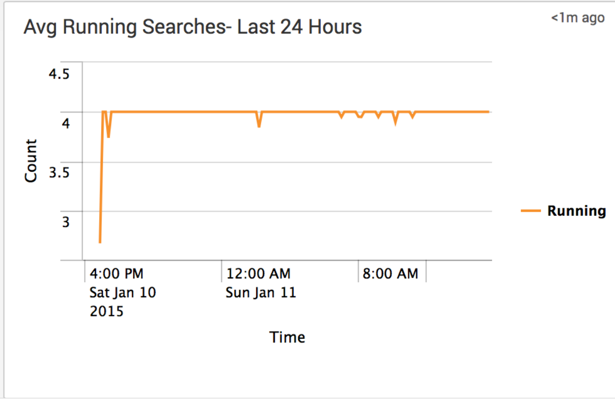
Splunk 6.3 Test Results

Things to notice:

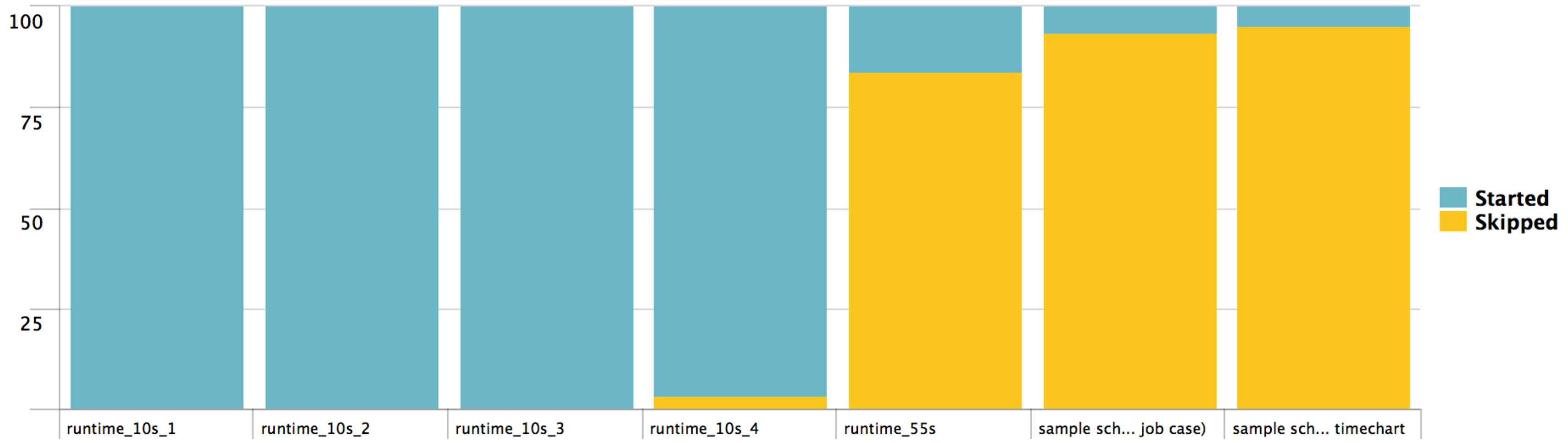
- Started/skipped forms less of a pattern: the search order is being perturbed
- From 12–2am, fewer searches are skipped
- Average running searches is mostly constant







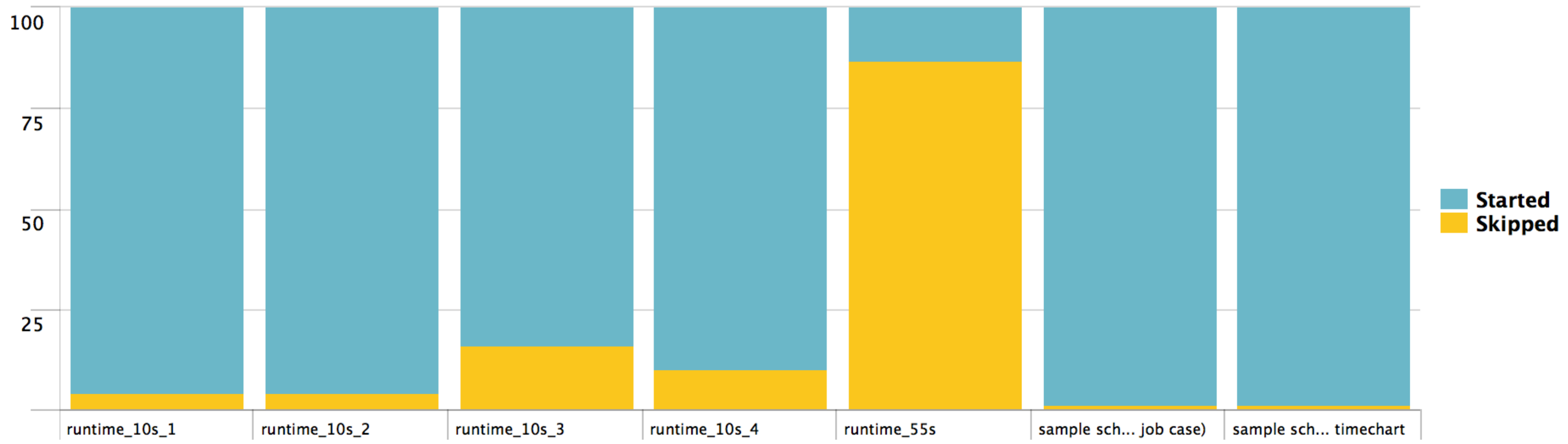
Splunk 6.2 Test Results



Things to notice:

- Number skipped gets worse in the order as the searches are in savedsearches.conf
- The "sample" searches, despite being very short, are almost always skipped

Splunk 6.3 Test Results



Things to notice:

- Number skipped is much more evenly distributed regardless of savedsearches.conf
- The "sample" searches are now almost never skipped (which is the sensible thing to do)

Takeaways

- Splunk Enterprise 6.3 adds better *priority scoring* and *search windows* for much improved saved search scheduling by at least 25%
- For infrequent searches (hourly, daily, etc.) use *schedule windows*
- Use the built-in scheduler performance reports (under *Activity* > *System Activity* > *Scheduler*) to monitor performance: lots of skipped searches or high lag is bad
- If, despite tuning, you still have frequently skipped searches or high lag, then you probably need a bigger CPU or more nodes in your cluster

.conf2015

Questions

splunk>

THANK YOU