Best practices for Forwarder Hierarchies

Richard Morgan
Principal Architect | Splunk

splunk> .conf19

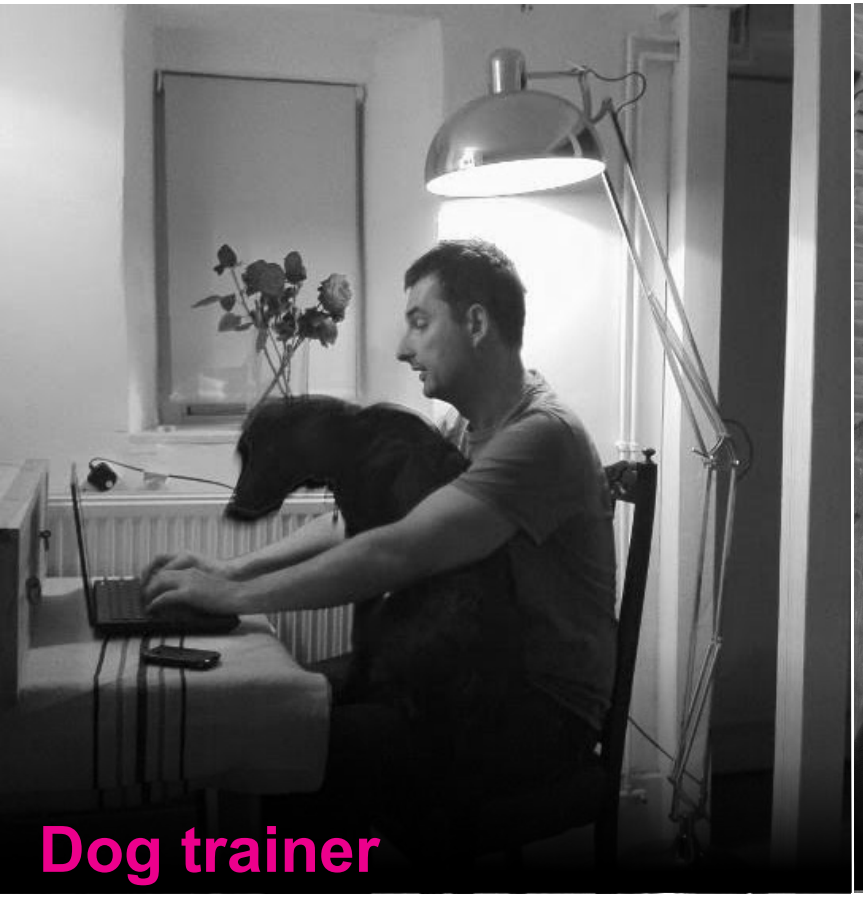
Forward-Looking Statements

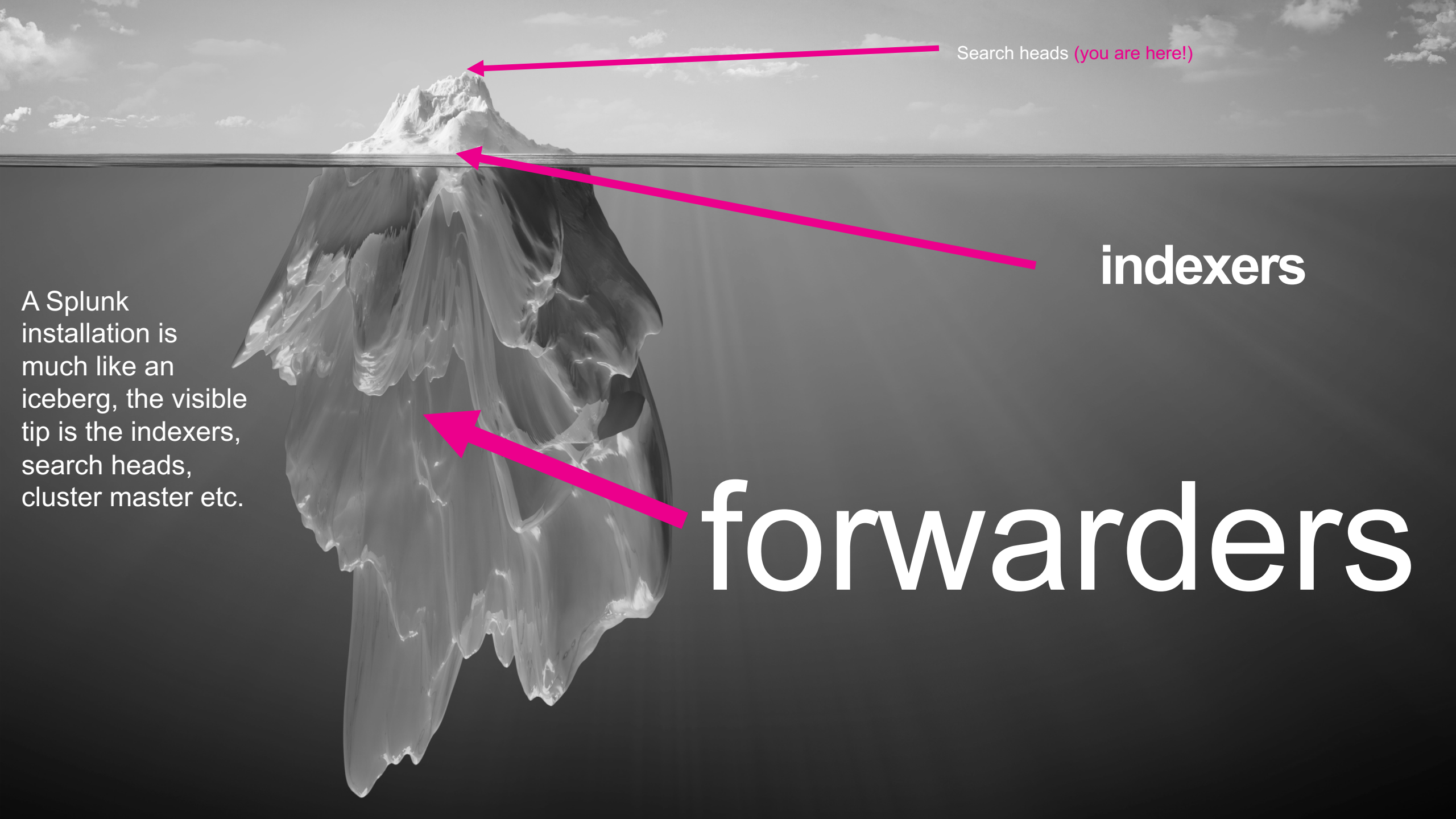
During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. 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.

Spreading that Splunk across EMEA since 2013

Self professed data junkie and SPL addict





A Splunk installation is much like an iceberg, the visible tip is the indexers, search heads, cluster master etc.

Why is event collection tuning important?

Data collection is the foundation of any Splunk instance

Event distribution underpins linear scaling of indexing and search

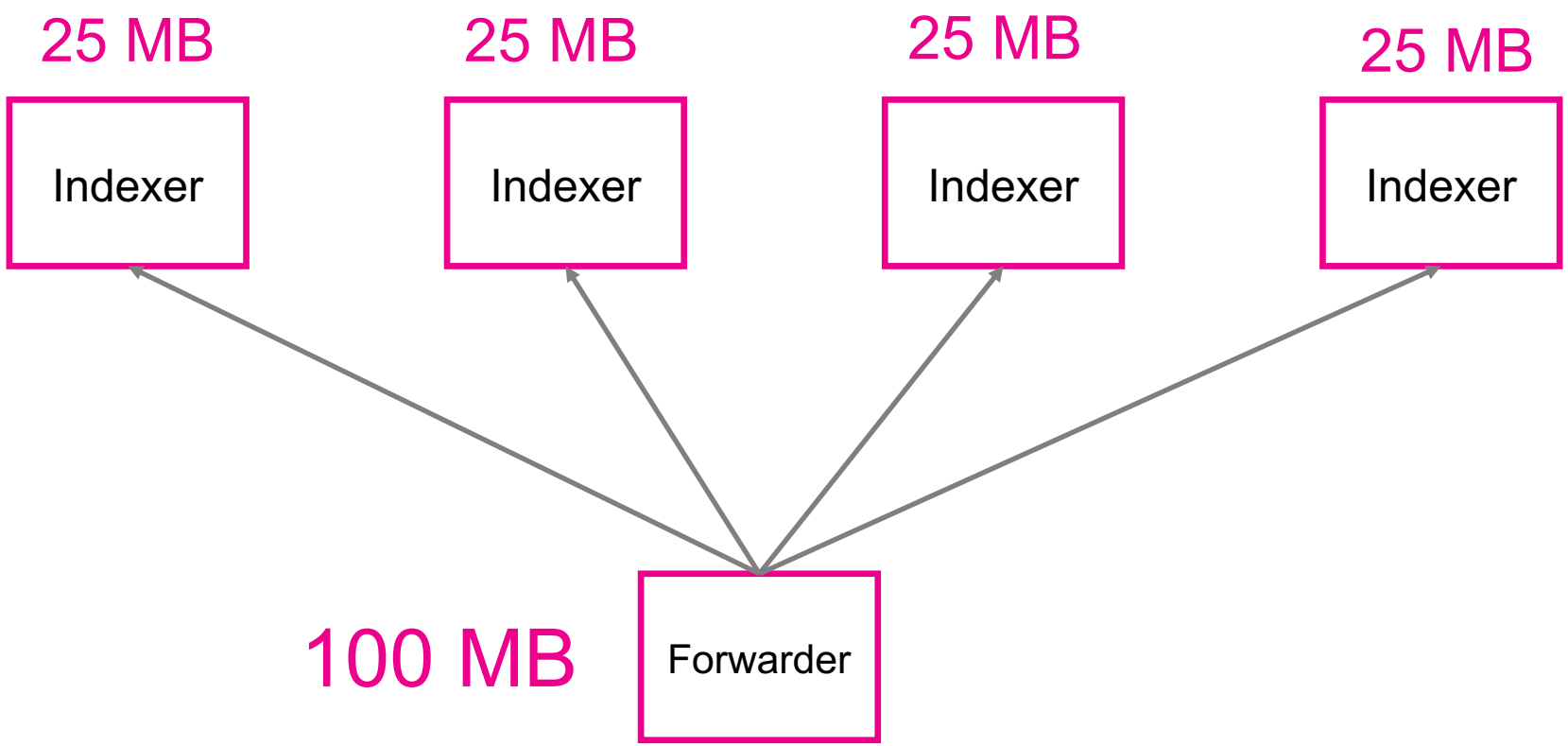
Events must be synchronized in time to corelate across hosts

Events must arrive in a timely fashion for alerts to be effective

What is event distribution?

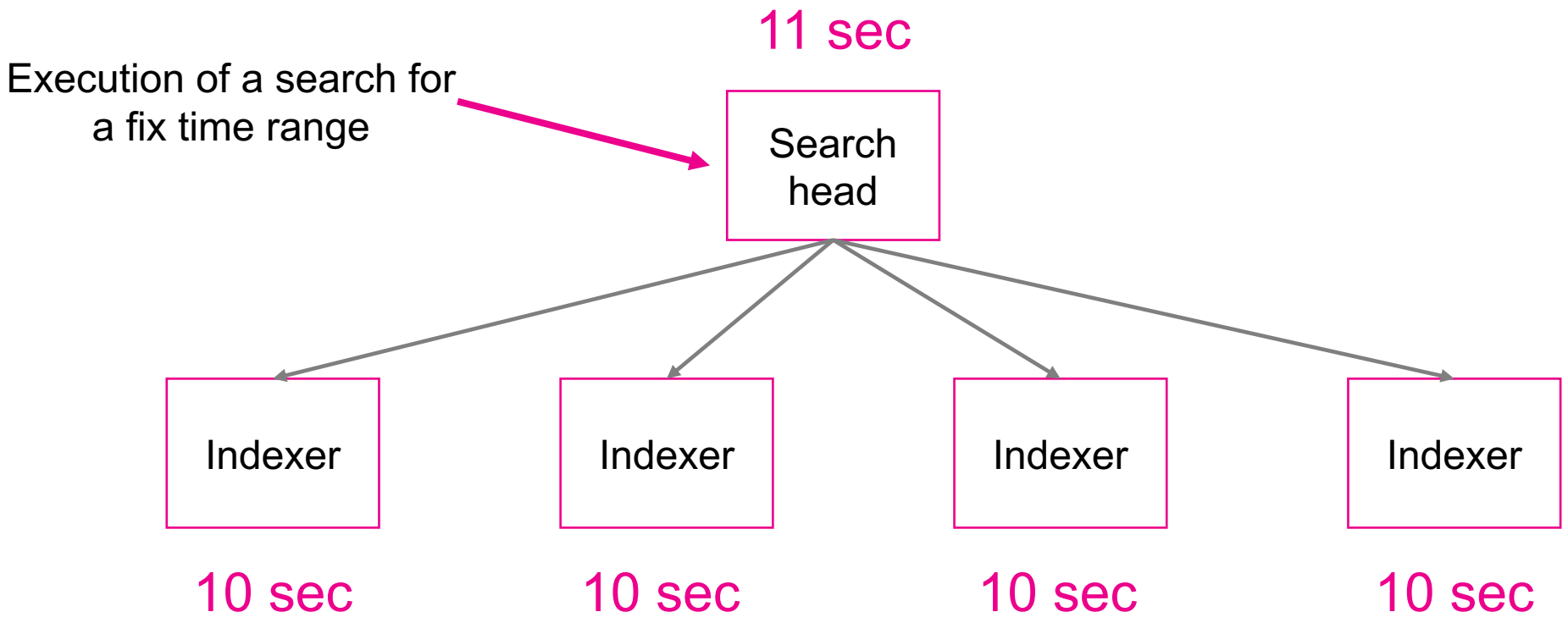
splunk> .conf19

What is Good Event Distribution?



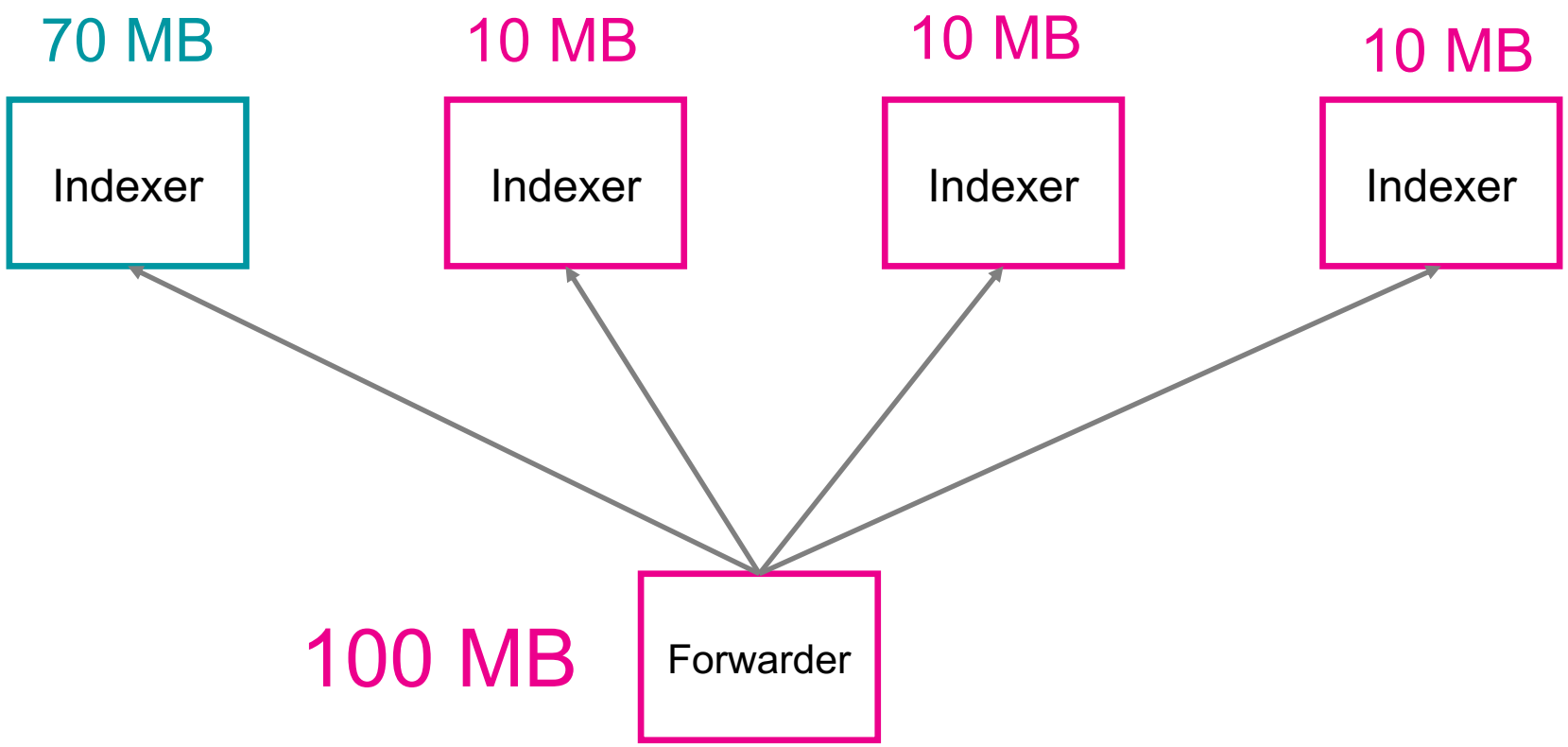
Event distribution is how Splunk spreads its incoming data across multiple indexers

Why is Good Event Distribution important?



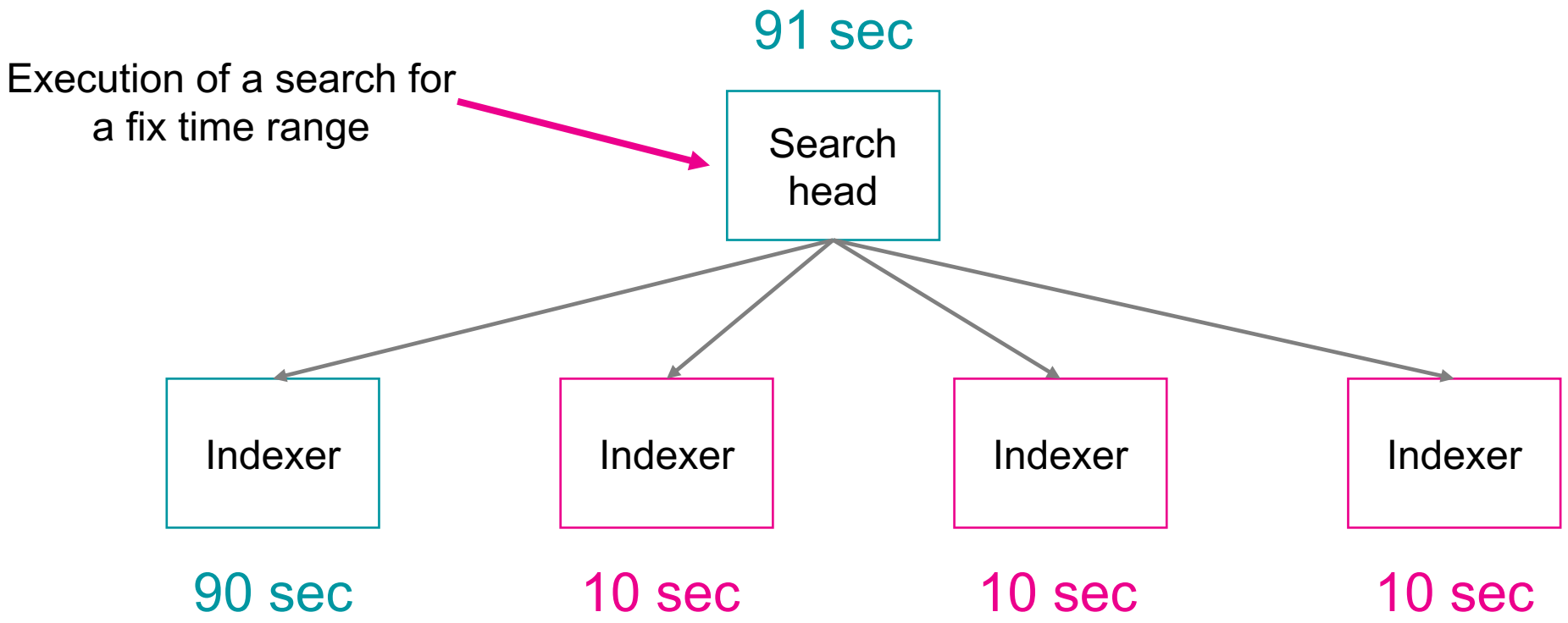
Event distribution is critical for the even distribution of search (computation) workload

What is ‘bad’ Event Distribution?



Bad event distribution is when the spread of events is **uneven** across the indexers

Bad Event Distribution affects search

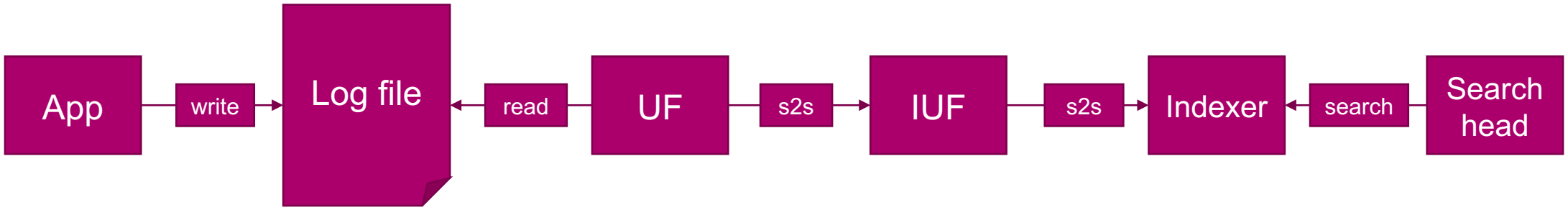


Search time becomes unbalanced, searches take longer to complete and reducing throughput

What is event delay?

splunk> .conf19

What is event delay?



The app needs to write out log events, it creates a buffer and then flushes it

The buffer is appended to the log file and the operating system notifies the UF of the change.

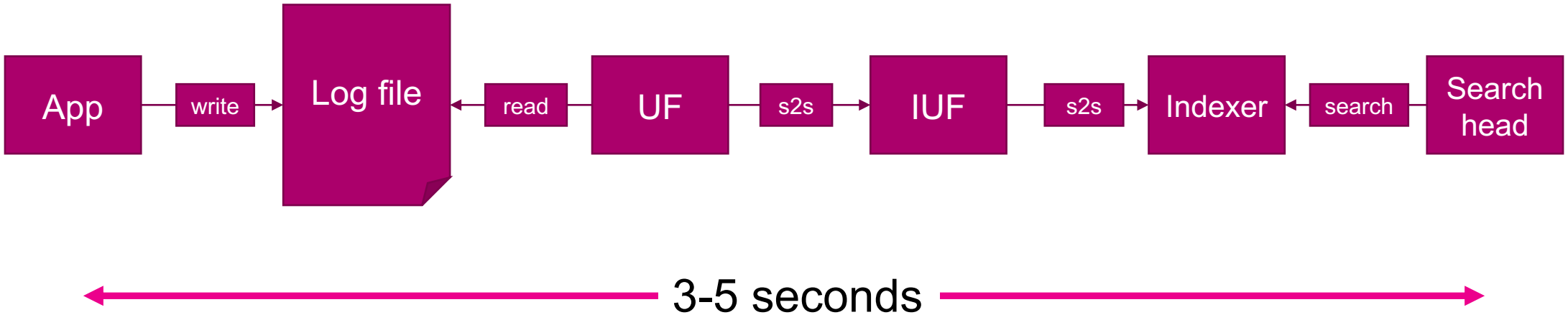
The delta is read, split up into 64kb blocks, compressed and transmitted with some additional meta data

Decodes S2S, processes, routes, encodes S2S, retransmits **streams** of data

Decodes S2S parses events from streams, via line breakers, and time extraction. Creates indexes and writes to disk

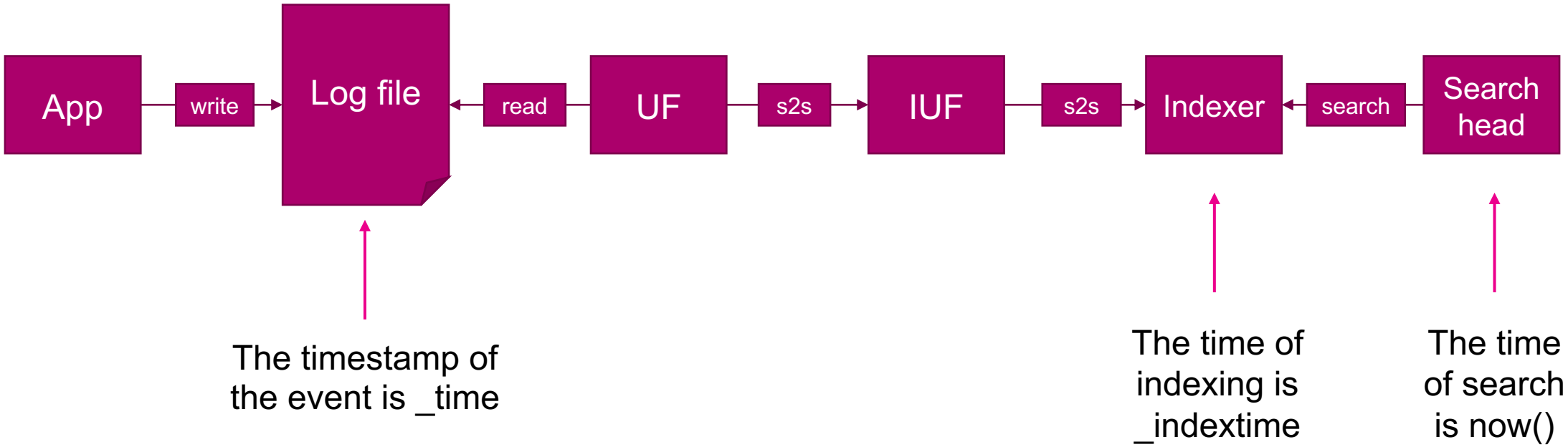
Sends search request to indexers, which open buckets files and and gets results returns them SH

What is good event delay?



It should take between 3-5 seconds from event generation to that event being searchable

How do you calculate event delay?




```
| eval event_delay=_indextime - _time
```

How streams affect event distribution

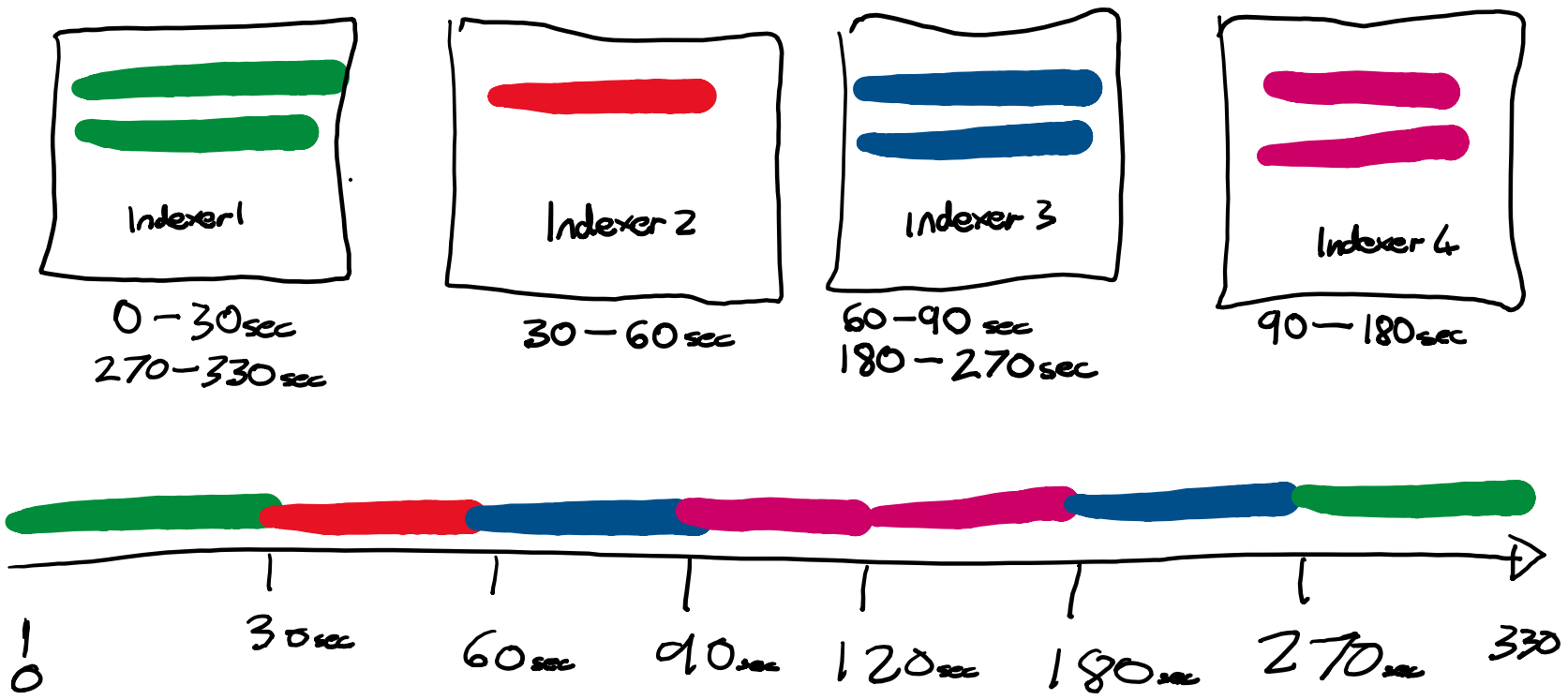
splunk> .conf19

autoLBFrequency (outputs.conf)

```
autoLBFrequency = <integer>
* The amount of time, in seconds, that a forwarder sends data to an indexer
  before redirecting outputs to another indexer in the pool.
* Use this setting when you are using automatic load balancing of outputs
  from universal forwarders (UFs).
* Every 'autoLBFrequency' seconds, a new indexer is selected randomly from the
  list of indexers provided in the server setting of the target group
  stanza.
* Default: 30
```

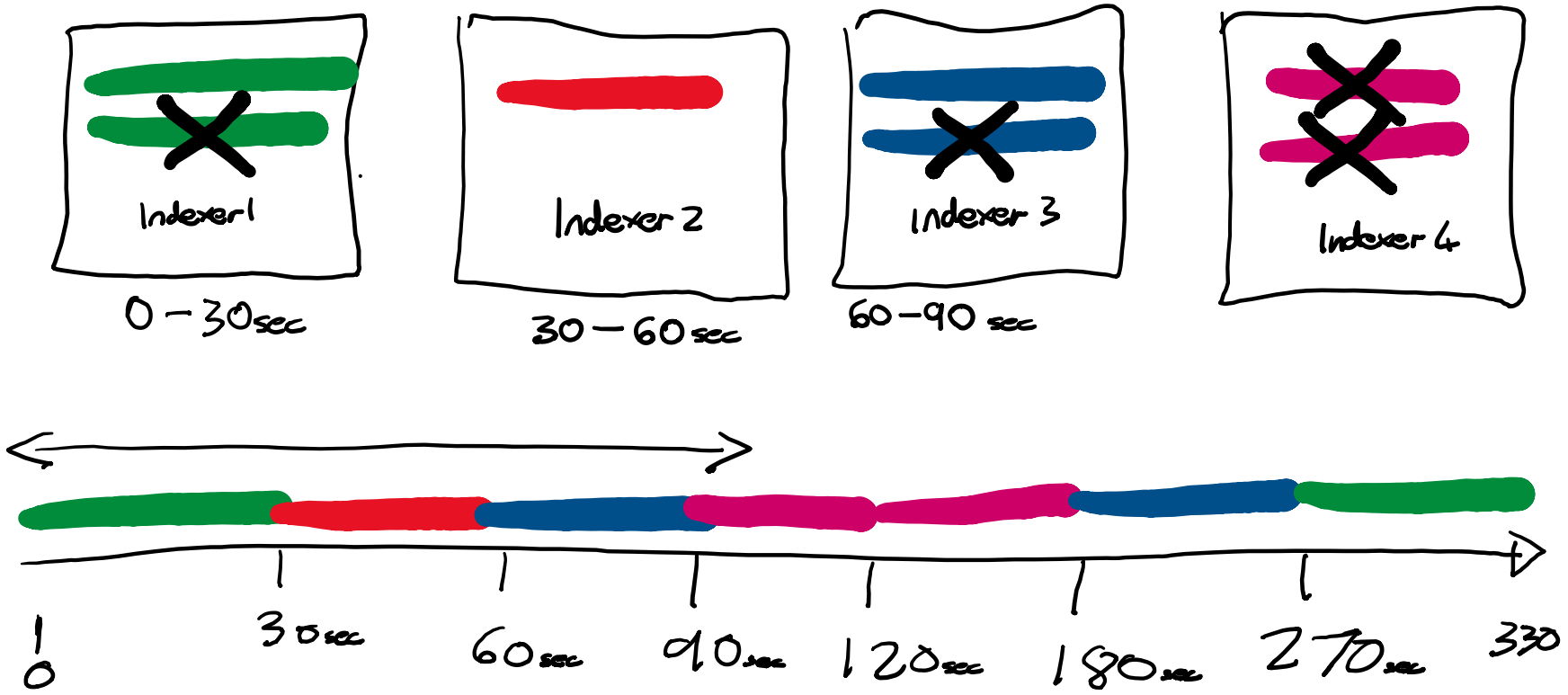
30 seconds of 1 MB/s is 30 MB for each connection!

Data is distributed across the indexers over time



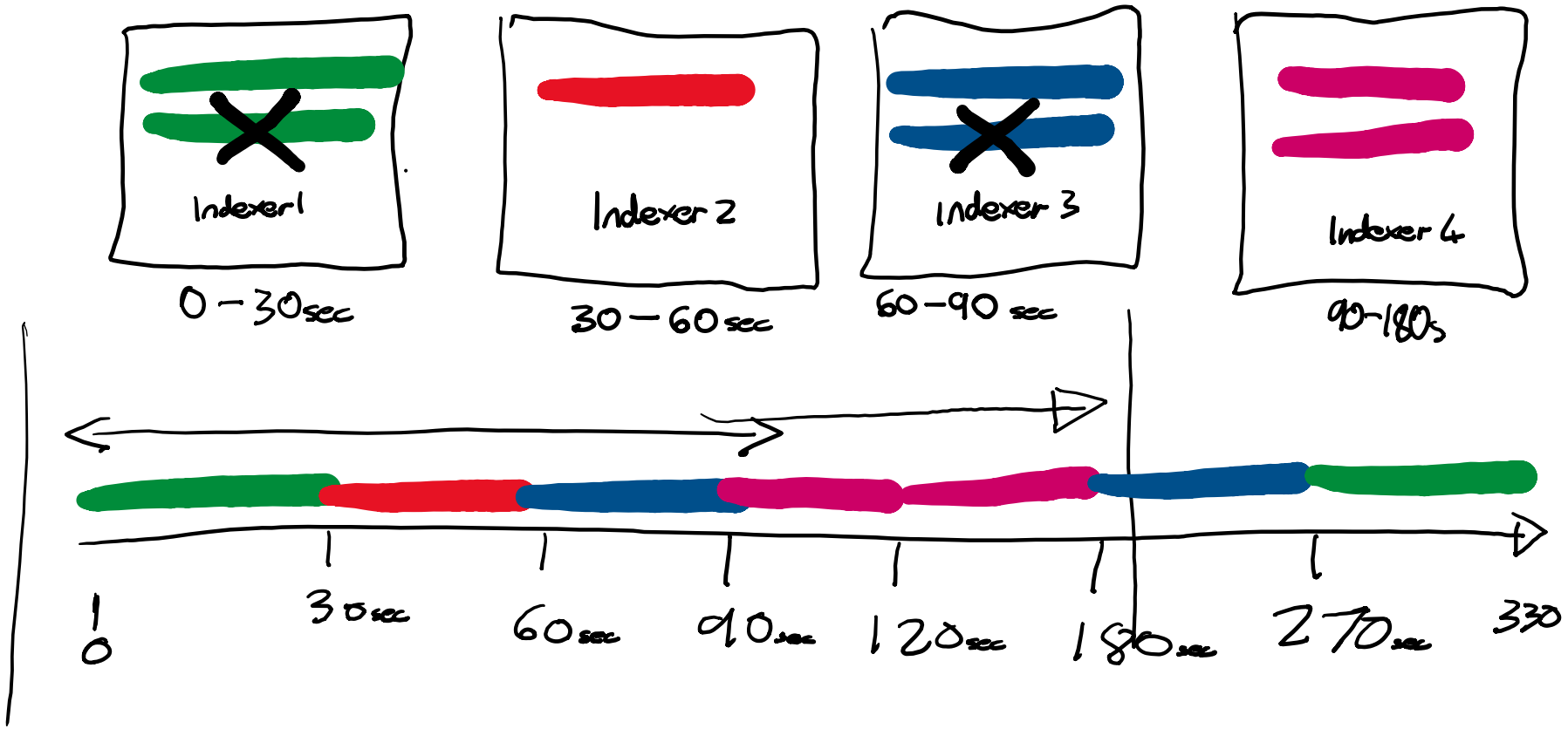
Each indexer is allocated 30s of data via Randomized Round Robin

earliest=now latest=-90s



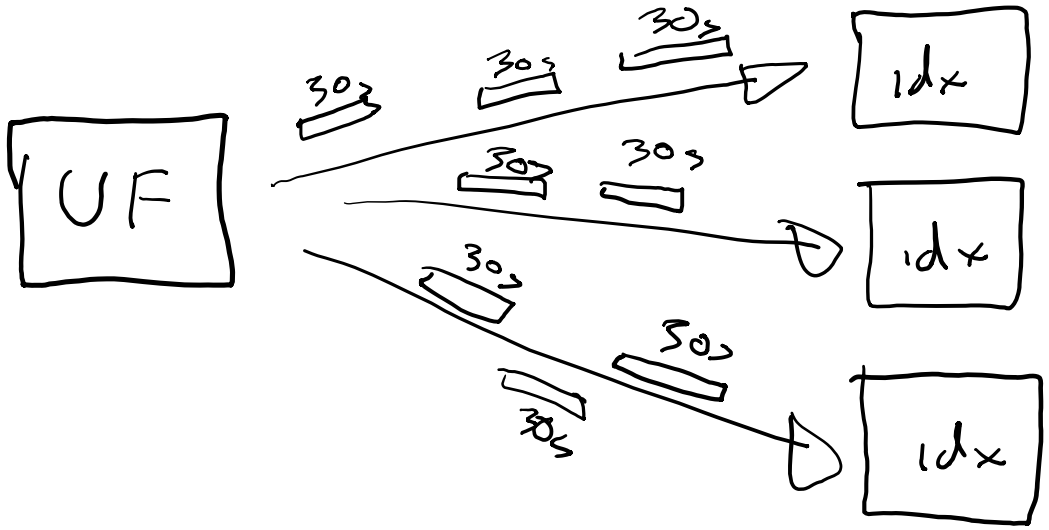
Indexer 4 has no data to process

earliest=now latest=-180s



Indexer 4 has 2x the data to process

Event distribution improves over time



Switching at 30 seconds

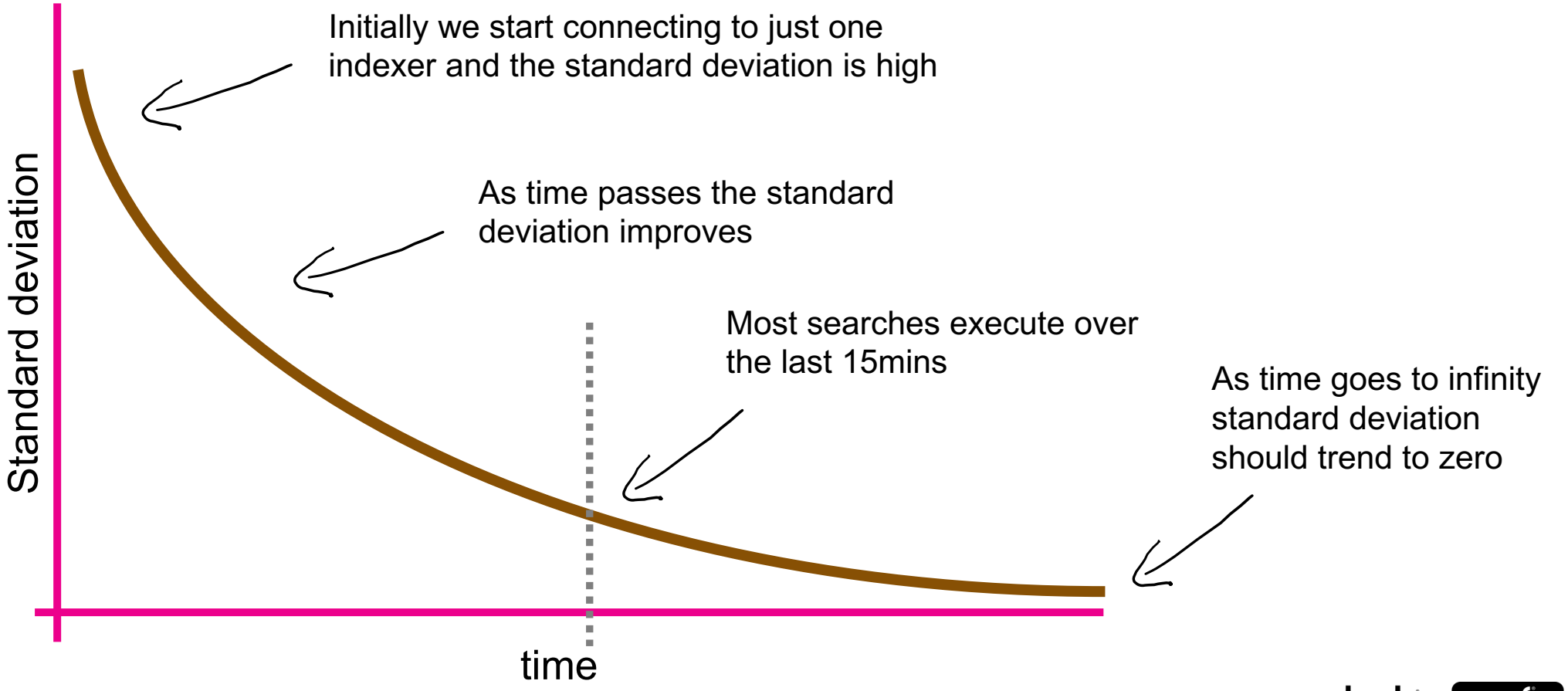
5 mins = 10 connections
1 hour = 120 connections
1 day = 2880 connections

How long before all indexers have data?

10 indexers = 5 mins
50 indexers = 25 mins
100 indexers = 50 mins

Larger clusters take longer to get “good” event distribution

Event distribution improves over time



It’s not Round Robin (RR), its Randomized RR

Round robin:

Round 1 order: 1,2,3,4

Round 2 order: 1,2,3,4

Round 3 order: 1,2,3,4

Round 4 order: 1,2,3,4

Randomized Round Robin:

Round 1 order: 1,4,3,2

Round 2 order: 4,1,2,3

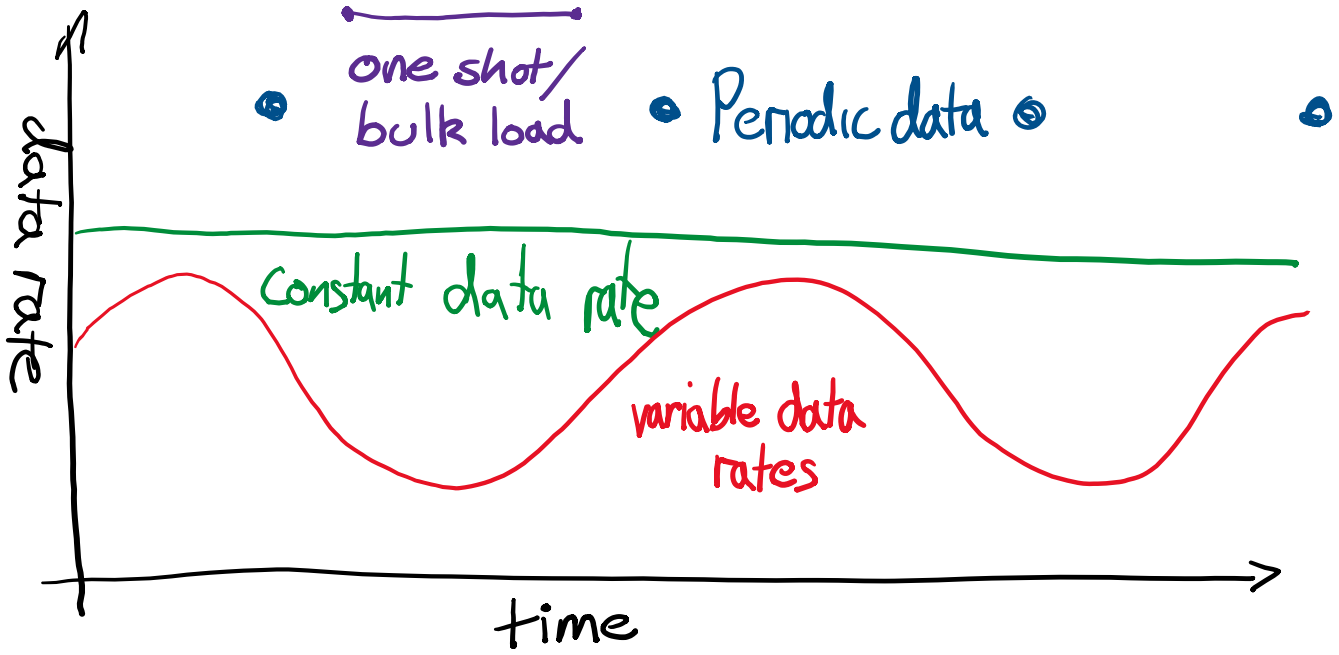
Round 3 order: 2,4,3,1

Round 4 order: 1,3,2,4

Splunk’s randomized round robin algorithm quickens event distribution

Types of data flows

Types of data flow coming into Splunk



Periodic data, typically a scripted input, very spikey. Think AWS CloudWatch, think event delay.

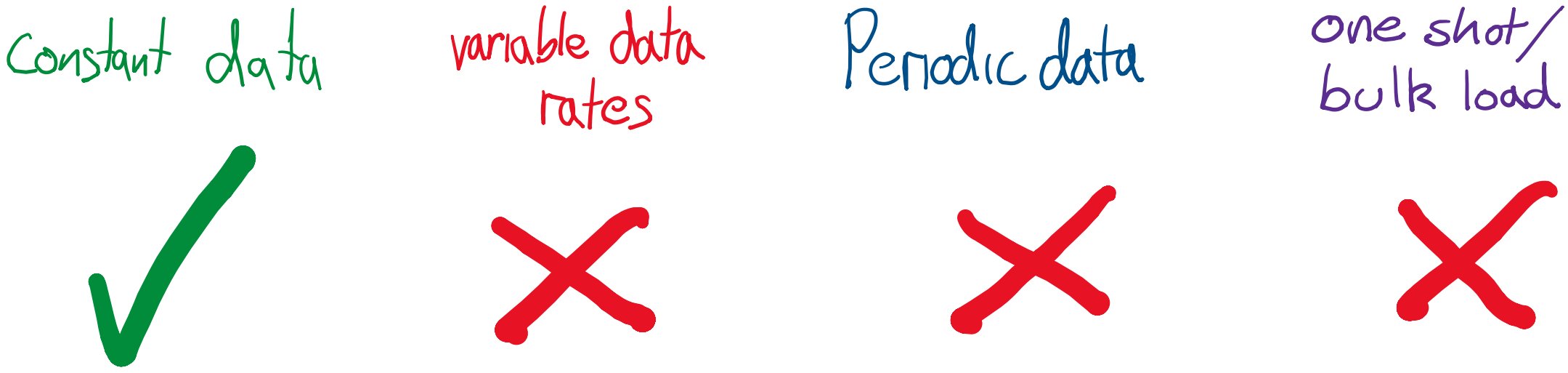
Constant data rates, nice and smooth, think metrics.

Variable data rates, typically driven by usage, think web logs

One shot, much like periodic data

It can be difficult to optimize for every type of data flow

Time based LB does not work well on its own



timebasedAutoLB is the default and is set to 30s

Best practice !!!

autoLBVolume (outputs.conf)

```
autoLBVolume = <integer>
* The volume of data, in bytes, to send to an indexer before a new indexer
  is randomly selected from the list of indexers provided in the server
  setting of the target group stanza.
* This setting is closely related to the 'autoLBFrequency' setting.
  The forwarder first uses 'autoLBVolume' to determine if it needs to switch to another
  indexer. If the 'autoLBVolume' is not reached,
  but the 'autoLBFrequency' is, the forwarder switches to another indexer as the
forwarding target.
* A non-zero value means that volume-based forwarding is active.
* 0 means the volume-based forwarding is not active.
* Default: 0
```

Switching too fast can result in lower throughput

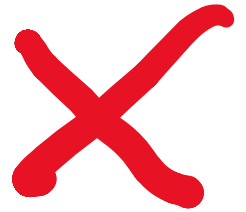
autoLBVolume + time based is much better

Constant data

variable data rates

Periodic data

one shot/ bulk load

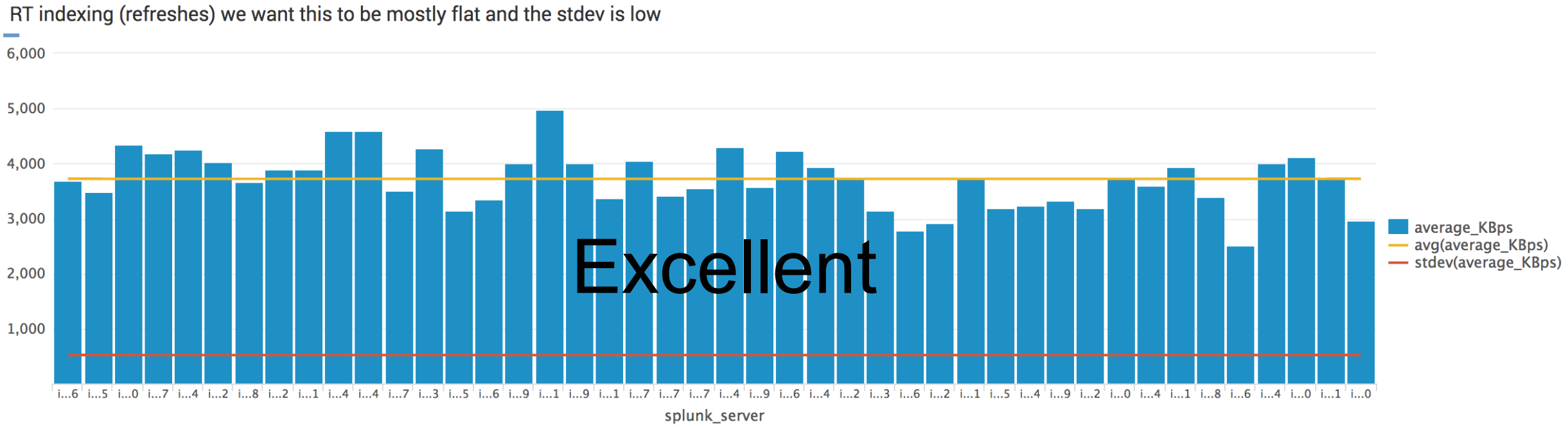


autoLBVolume is better for variable and bursty data flows

How to measure event distribution

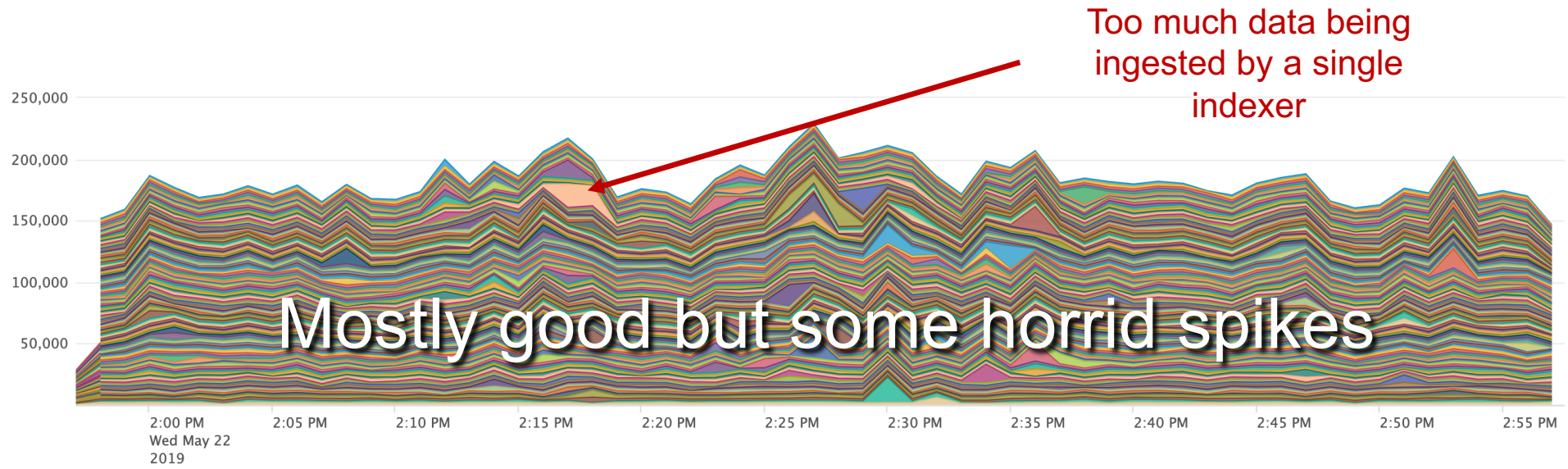
splunk> .conf19

Call the REST API to get RT ingestion




```
| rest /services/server/introspection/indexer
| eventstats stdev(average_KBps) avg(average_KBps)
```

Ingestion over time




```
index=_internal Metrics TERM(group=thruput) TERM(name=thruput)
sourcetype=splunkd
    [| dbinspect index=*
    | stats values(splunk_server) as indexer
    | eval host_count=mvcount(indexer),
        search="host IN (".mvjoin(mvfilter(indexer!=""), ", ").")"]
| eval host_pipeline=host."-".ingest_pipe
| timechart minspan=30sec limit=0 per_second(kb) by host_pipeline
```

splunk> .conf19

Indexer thruput search explained

Use metrics to get pipeline statistics

Use TERM to speed up search execution

```
index=_internal Metrics TERM(group=thruput) TERM(name=thruput)
sourcetype=splunkd
```

```
[| dbinspect index=*
| stats values(splunk_server) as indexer
| eval host_count=mvcount(indexer),
    search="host IN (".mvjoin(mvfilter(indexer!=""), ", ").")"]
```

```
| eval host_pipeline=host."-".ingest_pipe
| timechart minspan=30sec limit=0 per_second(kb) by host_pipeline
```

Measures each pipeline individually

Sub-search returns indexer list

Count the events per indexer via job inspector

dispatch.stream.remote	411	-	4,436,202
dispatch.stream.remote.idx-i-01706d9ff09694435.skynet.stg.splunkcloud.com	3	-	30,888
dispatch.stream.remote.idx-i-049ae5635d62c2fc2.skynet-search.splunkcloud.com	3	-	28,441
dispatch.stream.remote.idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2	-	22,159
dispatch.stream.remote.idx-i-093a387fcef103e13.skynet-virginia.splunkcloud.com	3	-	29,931
dispatch.stream.remote.idx-i-056433c5af11fc2aa.skynet-virginia.splunkcloud.com	3	-	32,545
dispatch.stream.remote.idx-i-0e51c7a51b02d8e9e.skynet-virginia.splunkcloud.com	3	-	39,127
dispatch.stream.remote.idx-i-04f26b6ff38aef567.skynet-virginia.splunkcloud.com	3	-	35,165
dispatch.stream.remote.idx-i-09a635cb80079cc63.skynet-virginia.splunkcloud.com	3	-	33,545
dispatch.stream.remote.idx-i-005155ae6b23ec367.skynet-virginia.splunkcloud.com	3	-	31,245
dispatch.stream.remote.idx-i-08040332e517b8b64.skynet-virginia.splunkcloud.com	3	-	33,281
dispatch.stream.remote.idx-i-05a2724c6185e8f65.skynet-virginia.splunkcloud.com	3	-	39,460
dispatch.stream.remote.idx-i-0fa5aa596[illegible]virginia.splunkcloud.com	3	-	32,234
dispatch.stream.remote.idx-i-03adf[illegible]skynet-virginia.splunkcloud.com	3	-	34,195
dispatch.stream.remote.idx-i-0da[illegible]5f8698c.skynet-virginia.splunkcloud.com	3	-	31,536
dispatch.stream.remote.idx-i-0547252[illegible]skynet-virginia.splunkcloud.com	3	-	34,531
dispatch.stream.remote.idx-i-05b3ad40289213160.skynet-virginia.splunkcloud.com	3	-	33,204
dispatch.stream.remote.idx-i-09225b916b0b[illegible]virginia.splunkcloud.com	3	-	32,146
dispatch.stream.remote.idx-i-085b2751f51b5e2e[illegible]skynet-virginia.splunkcloud.com	3	-	35,108
dispatch.stream.remote.idx-i-0f3bc76cbe02990ab.skynet-virginia.splunkcloud.com	3	-	39,725
dispatch.stream.remote.idx-i-03b2afa080d03f23c.skynet-virginia.splunkcloud.com	3	-	31,541
dispatch.stream.remote.idx-i-0f99a2913971e27a8.skynet-sydney.splunkcloud.com	3	-	30,838
dispatch.stream.remote.idx-i-0256979b3e6ec19d5.skynet-virginia.splunkcloud.com	3	-	30,230
dispatch.stream.remote.idx-i-03d736555c18dc8e5.skynet-infra.splunkcloud.com	3	-	31,112
dispatch.stream.remote.idx-i-0ac63edf1bfbbbf59.skynet-oregon.splunkcloud.com	3	-	34,313

Nice and healthy

For any search, open the job inspector and see how many events each indexer returned.

This should be approximately the same for each indexer.

In a multi site or multi cluster environment we expect that groups of indexers are likely to have different numbers of events.

Within a single site the number of events should be the about same for each indexer.

Count the events per indexer and index via search

New Search

Save As | New Table | Close

```
| tstats count where
    index=*
    splunk_server=*
    host=* earliest=-5min latest=now
    by splunk_server index
| stats
    sum(count) as total_events
    stdev(count) as stdev
    avg(count) as average
    by index
| eval normalized_stdev=stdev/average
```

Last 30 minutes

The event distribution score per index

✓ 235,208,868 events (Partial results for 6/12/19 1:22:56.000 PM to 6/12/19 1:27:56.000 PM) No Event Sampling | Job | Verbose Mode

Events | Patterns | Statistics (61) | Visualization

100 Per Page | Format | Preview

index	total_events	stdev	average	normalized_stdev
customer_splunkd	107336154	384942.7654871033	1277811.357142857	0.3012516388552238
customer_metrics	39908224	200352.08397764483	518288.6233766234	0.3865646956947684
customer_security	25396729	93304.89581460018	279084.93406593404	0.3343243737856406
aws_s3	12200913	1820179.9463761377	1525114.125	1.193471306
aws_vpc_flow_log	10180585	655318.6417824184	1272573.125	0.5149555879
skynet_uf	8819276	42832.80601104157	100219.04545454546	0.4273918776293721

Calculate event distribution per index for -5mins

```
| tstats count where
    index=*
    splunk_server=*
    host=* earliest=-5min latest=now
    by splunk_server index
| stats
    sum(count) as total_events
    stdev(count) as stdev
    avg(count) as average
    by index
| eval normalized_stdev=stdev/average
```

Narrow indexes

Narrow to site or cluster

Calculate per index, or host

Calculate score

Event distribution can be measured per index and per host

Shameless self promotion

Use my event distribution measurement dashboard to assess your stack

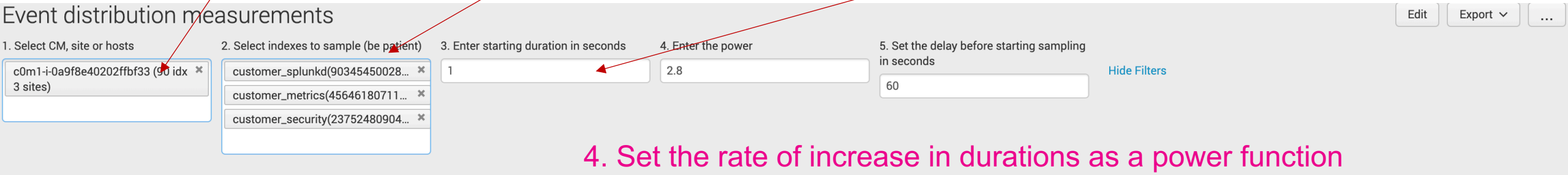
http://bit.ly/2WxRXvI

How to configure the event distribution dashboard

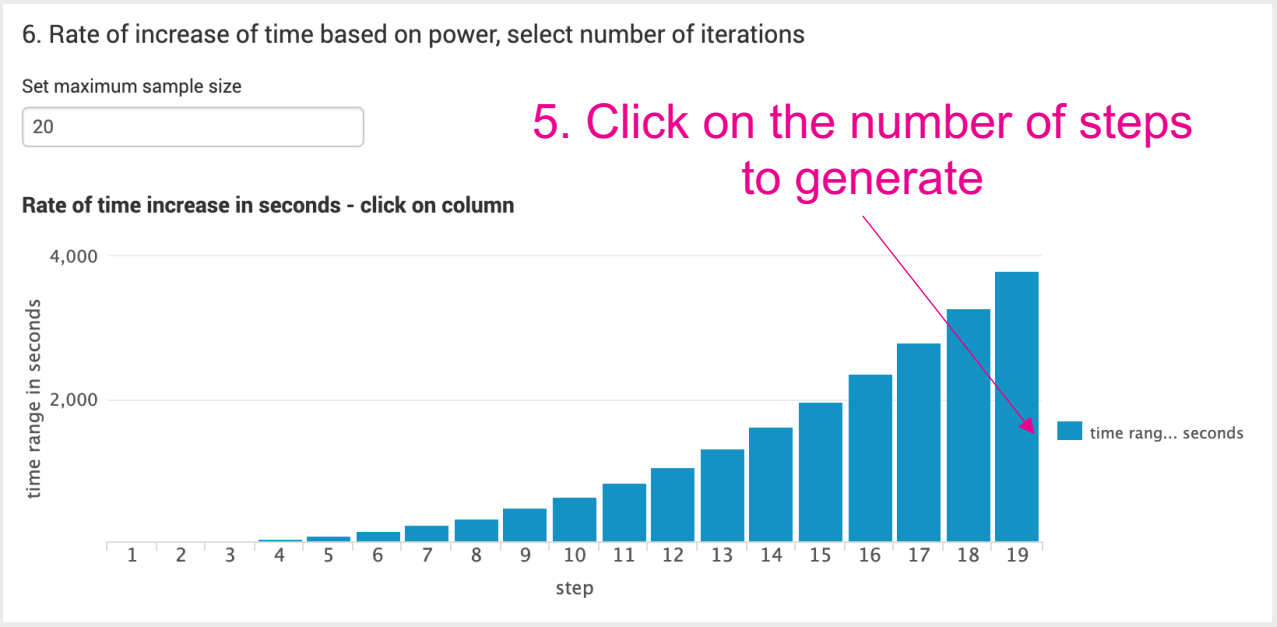
1. Select the site or cluster to analyze

2. Select the indexes present in the selected site or cluster

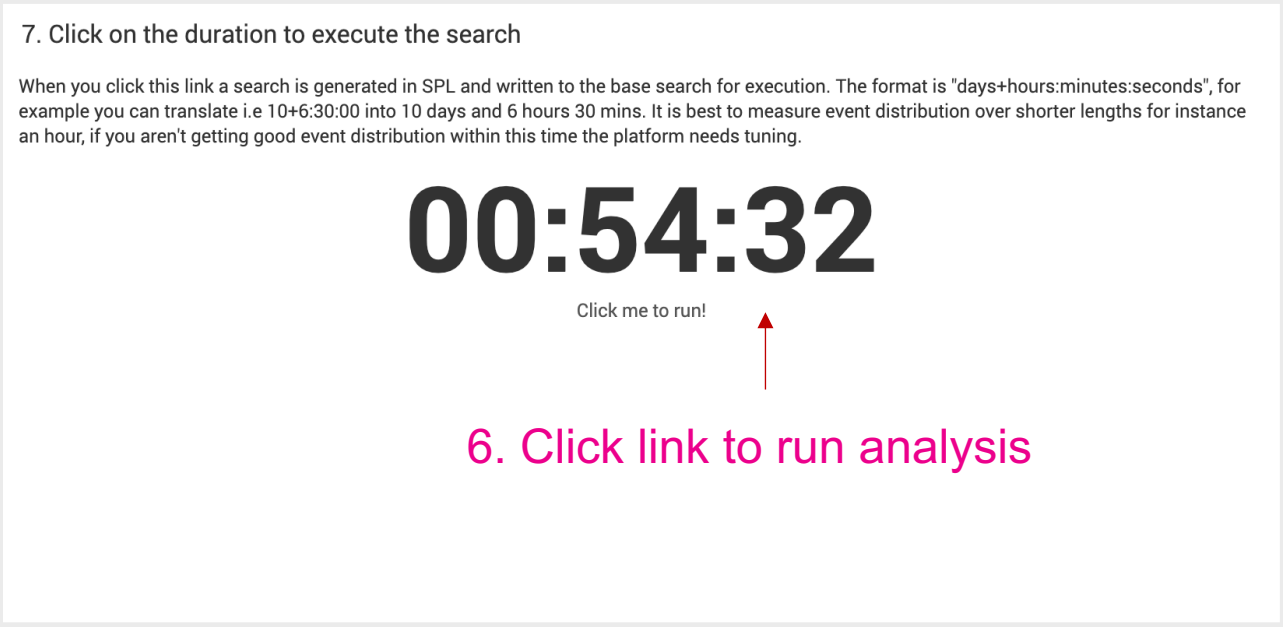
3. Set starting duration, one second is the default



4. Set the rate of increase in durations as a power function



5. Click on the number of steps to generate



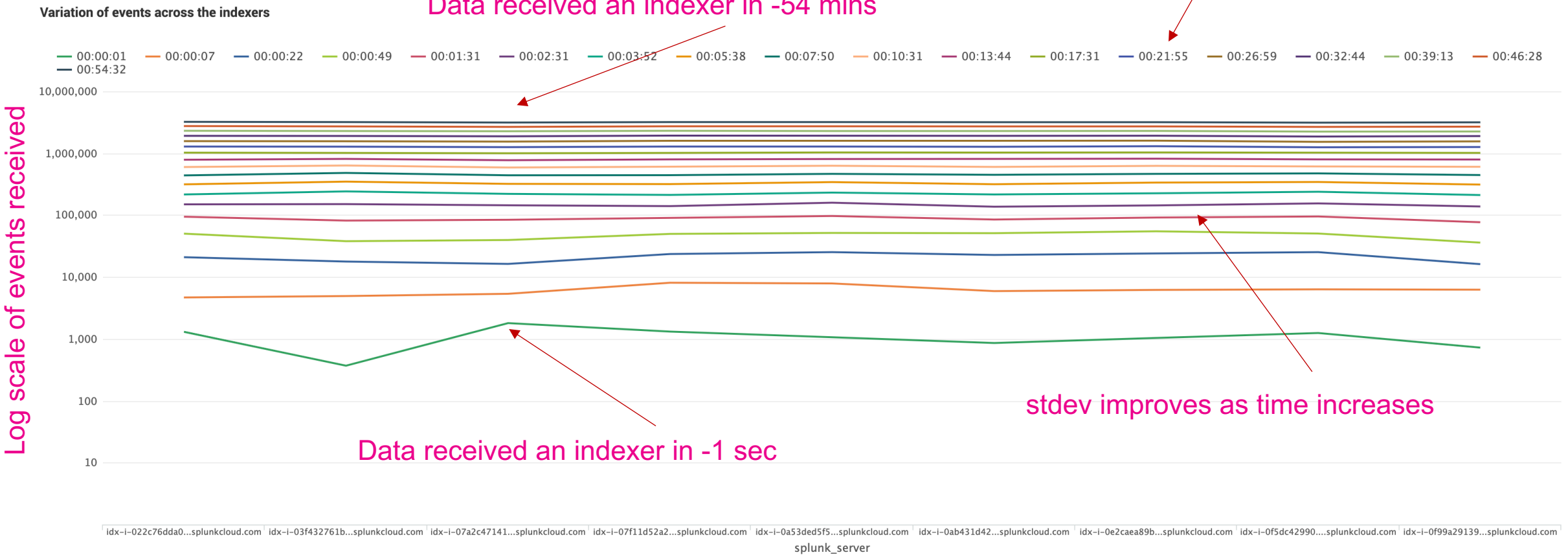
6. Click link to run analysis

How to read the first panel

Variation of events across the indexers

Each series is a time range, exponentially growing.

Data received an indexer in -54 mins



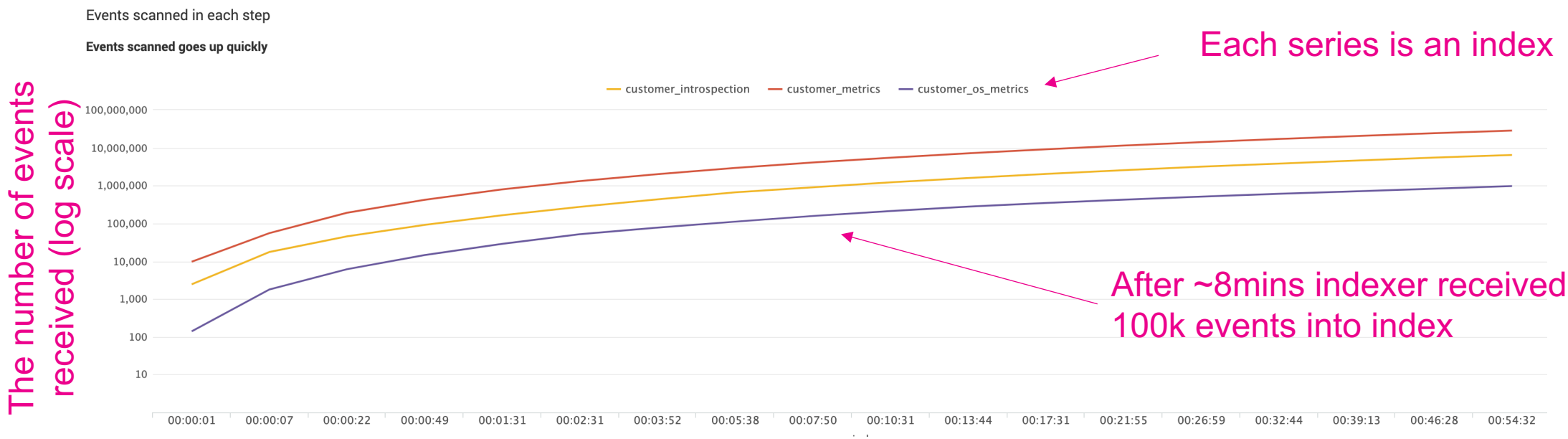
Data received an indexer in -1 sec

stdev improves as time increases

Each column is an indexer

How to read the second panel

Events scanned in each step



Each series is an index

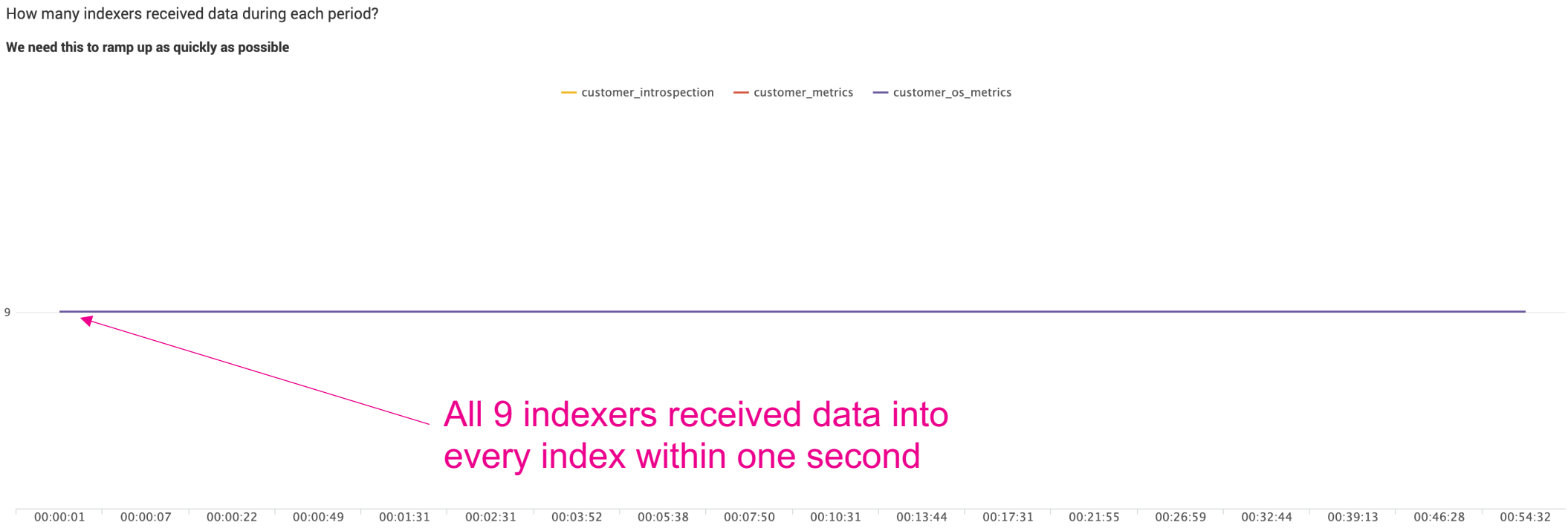
After ~8mins indexer received 100k events into index

Each time series plotted on x axis

splunk> .conf19

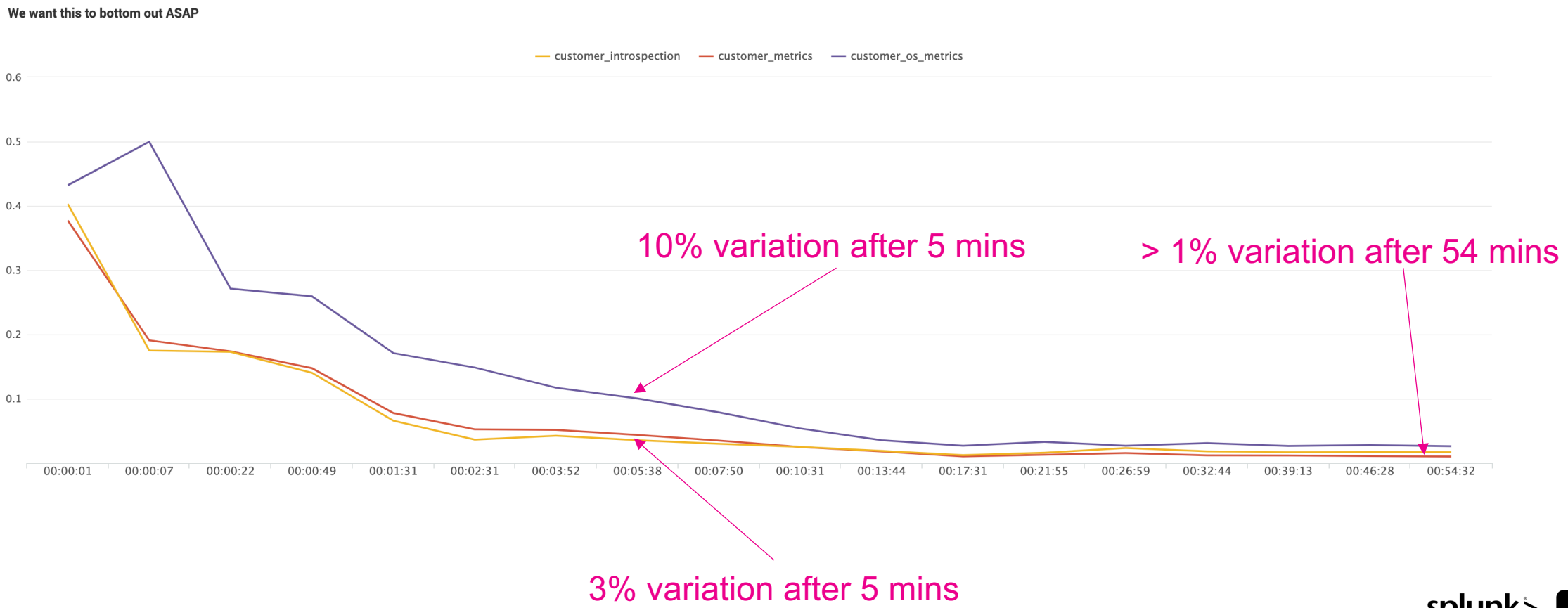
How to read the third panel

How many indexers received data in time range



How to read the final panel

How event distribution is improving over time

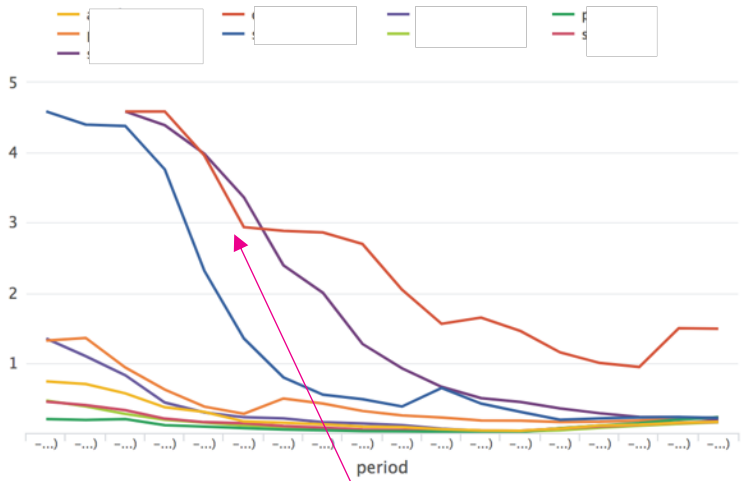


splunk> .conf19

You are viewing 18 steps starting with a period of 1 seconds increasing at the power 4.2 and spliting the data by index

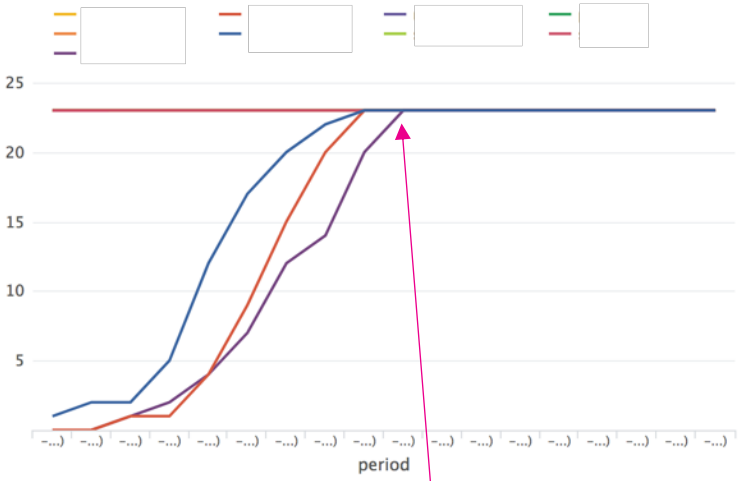
What was the normalised standard deviation over time?

We want this to bottom out ASAP



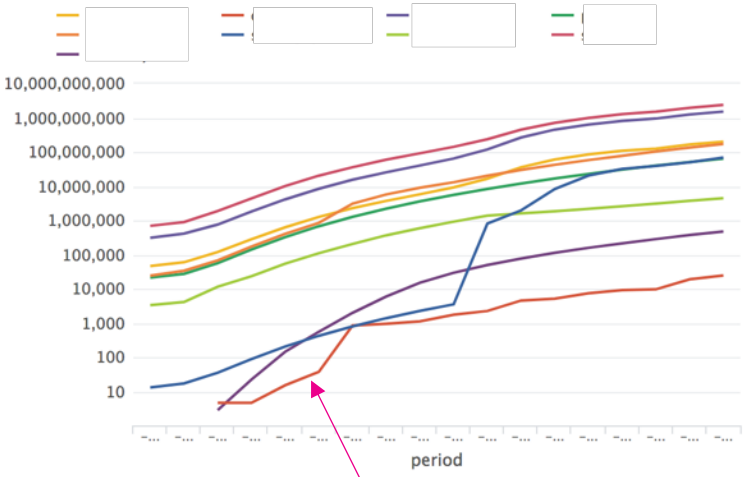
How many indexers received data during each period?

We need this to ramp up as quickly as possible



Events scanned in each step

Events scanned goes up quickly



Randomization is not happening fast enough

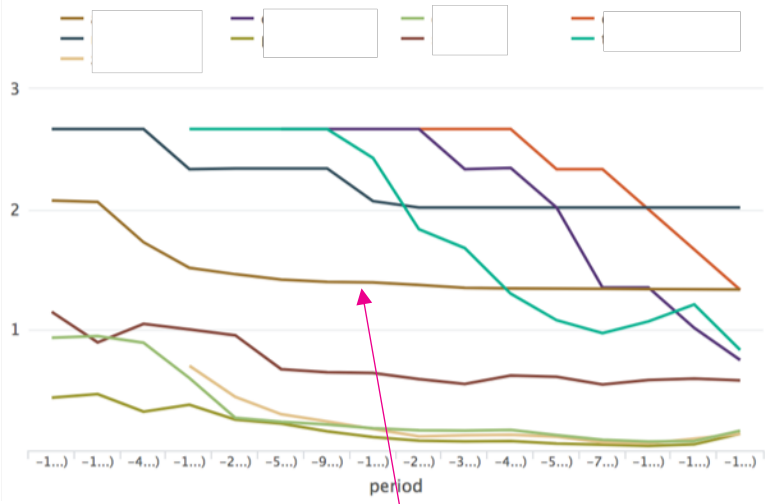
Three hours before all indexers have data, need to accelerate to improve entropy

Incoming data rates for indexers very different for the different indexes

splunk> .conf19

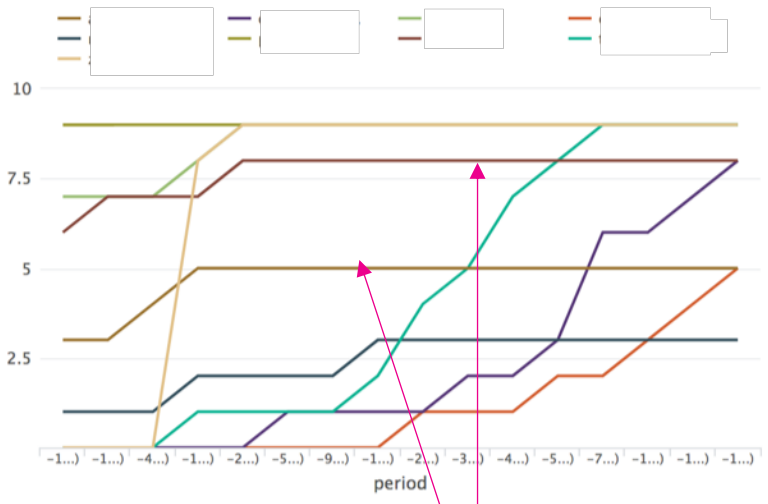
What was the normalised standard deviation over time?

We want this to bottom out ASAP



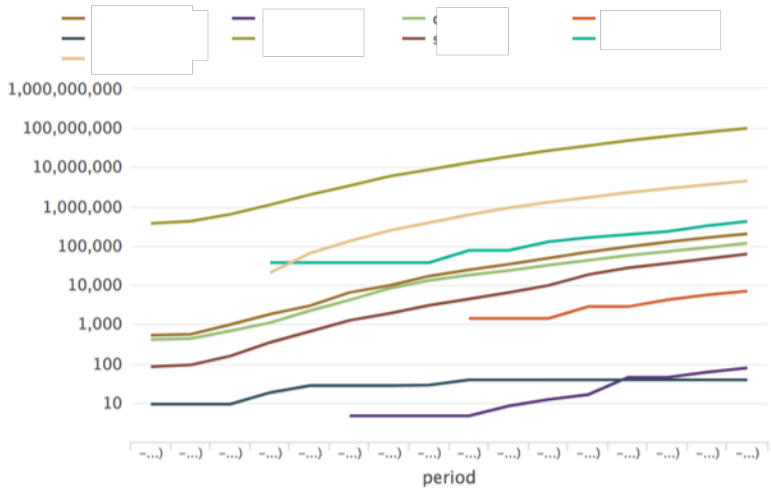
How many indexers received data during each period?

We need this to ramp up as quickly as possible



Events scanned in each step

Events scanned goes up quickly



Randomization is slow for most indexes and plateaus for some

Some indexers are not getting data, likely misconfigured forwarders

splunk> .conf19

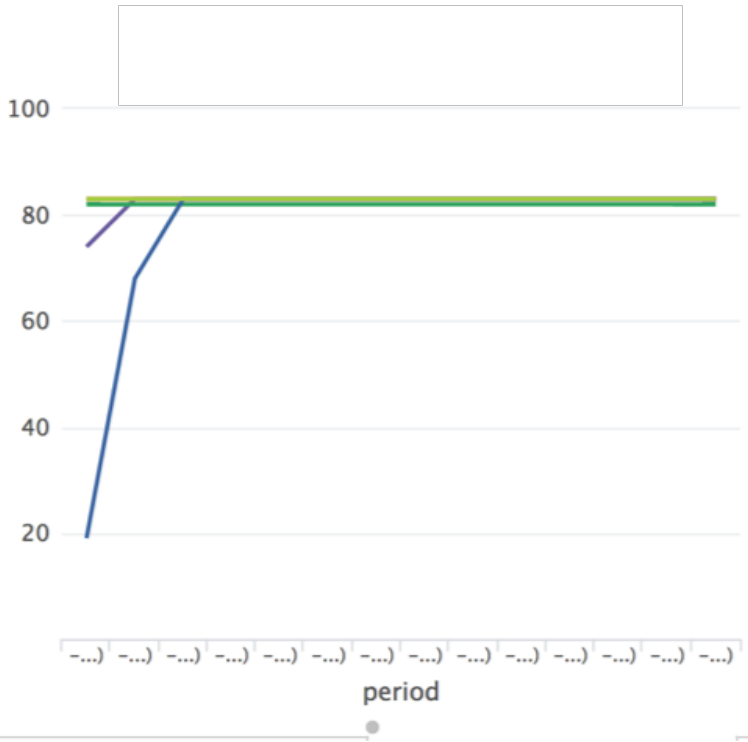
Near perfect event distribution using intermediate forwarders

What was the normalised standard deviation over time?

Until all indexers have received data the standard deviation cannot be calculated

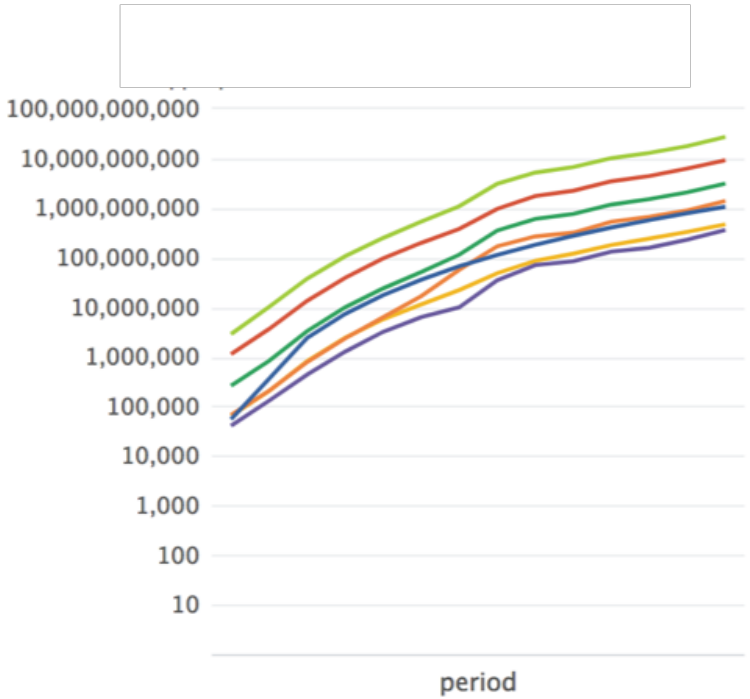
How many indexers received data during each period?

We need this to ramp up as quickly as possible



Events scanned in each step

Events scanned goes up quickly



Measuring event delay

Use TSTATS to compute event delay at scale

```
| tstats max(_indextime) as indexed_time count where index=*
    latest=+1day earliest=-1day _index_latest=-1sec _index_earliest=-2sec
    by index host splunk_server _time span=1s
| eval _time=round(_time), delay=indexed_time-_time,
    delay_str=tostring(delay,"duration")
| eventstats max(delay) as max_delay
    max(_time) as max_time
    count as eps
    by host index
| where max_delay = delay
| eval max_time=_time
| sort - delay
```

Last 30 minutes

✓ 226,034 events (Partial results for 6/11/19 2:08:30.000 PM to 6/13/19 2:08:30.000 PM) No Event Sampling Job Verbose Mode

Events Patterns Statistics (3,228) Visualization

100 Per Page Format Preview

< Prev 1 2 3 4 5 6 7 8 9 ... Next >

index	host	splunk_server	_time	indexed_time	count	delay	delay_str	eps	max_delay
customer_			2019-06-12 08:59:11	1560344908	16	18557	05:09:17	1	18557
customer_			2019-06-12 10:58:28	1560344908	43	11400	03:10:00	1	11400
customer_			2019-06-12 11:16:23	1560344908	351	10325	02:52:05	2	10325
customer_			2019-06-12 11:35:43	1560344908	27	9165	02:32:45	2	9165
customer_			2019-06-12 12:03:14	1560344908	4	7514	02:05:14	5	7514
customer_			2019-06-12 13:02:51	1560344908	1	3937	01:05:37	7	3937

splunk> .conf19

Delay per index and host

Use tstats because indextime as an indexed field!

```
| tstats max(_indextime) as indexed_time count
where index=*
    latest=+1day earliest=-1day
_index_latest=-1sec _index_earliest=-2sec
    by index host splunk_server _time span=1s
| eval _time=round(_time), delay=indexed_time-
_time,
    delay_str=tostring(delay,"duration")
| eventstats max(delay) as max_delay
    max(_time) as max_time
    count as eps
    by host index
| where max_delay = delay
| eval max_time=_time
| sort - delay
```

Delay per index and host

```
| tstats max(_indextime) as indexed_time count where index=*
    latest=+1day earliest=-1day _index_latest=-1sec _index_earliest=-2sec
    by index host splunk_server _time span=1s
| eval _time=round(_time), delay=indexed_time-_time,
    delay_str=tostring(delay,"duration")
| eventstats max(delay) as max_delay
    max(_time) as max_time
    count as eps
    by host index
| where max_delay = delay
| eval max_time=_time
| sort - delay
```

max(_indextime) is latest event

Split by _time to a resolution of a second

Compute the delta

Drop subseconds

Get events received in the last second, irrespective of the delay

More shameless self promotion

I have a dashboards that use this method to measure delay

http://bit.ly/2R9i4Yx

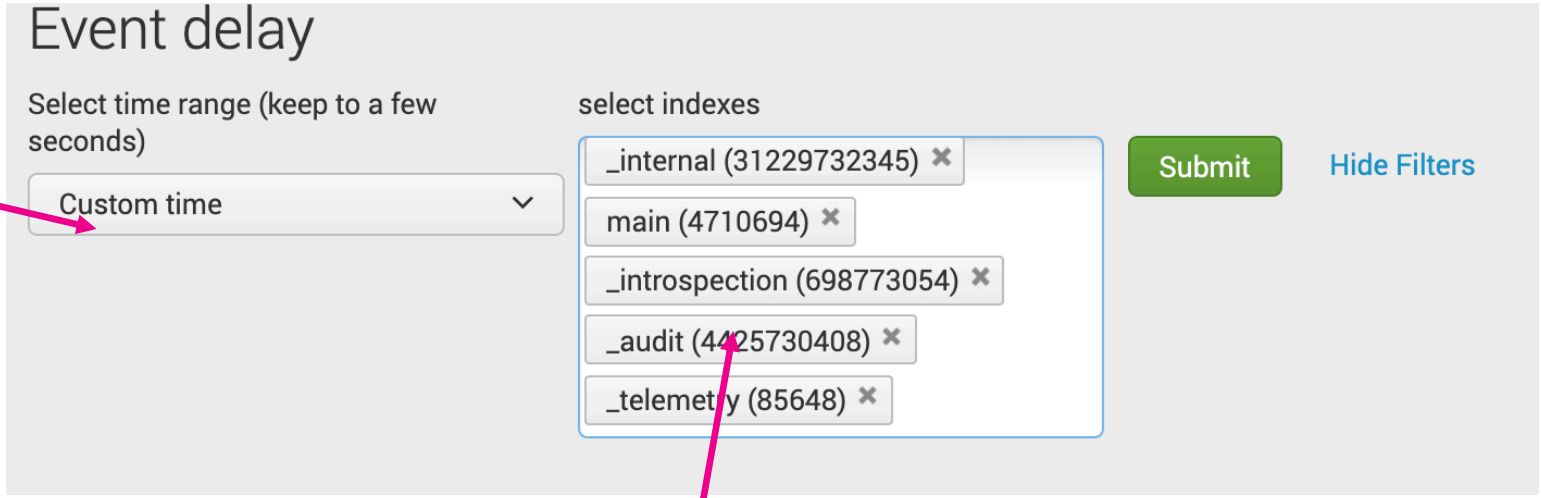
http://bit.ly/2I8qtIS

How to use the event delay dashboard

Drills down to per host delay

Keep time range short, but extend to capture periodic data

translates into _indextime, not _time



Select indexes to measure

How to read the main panels

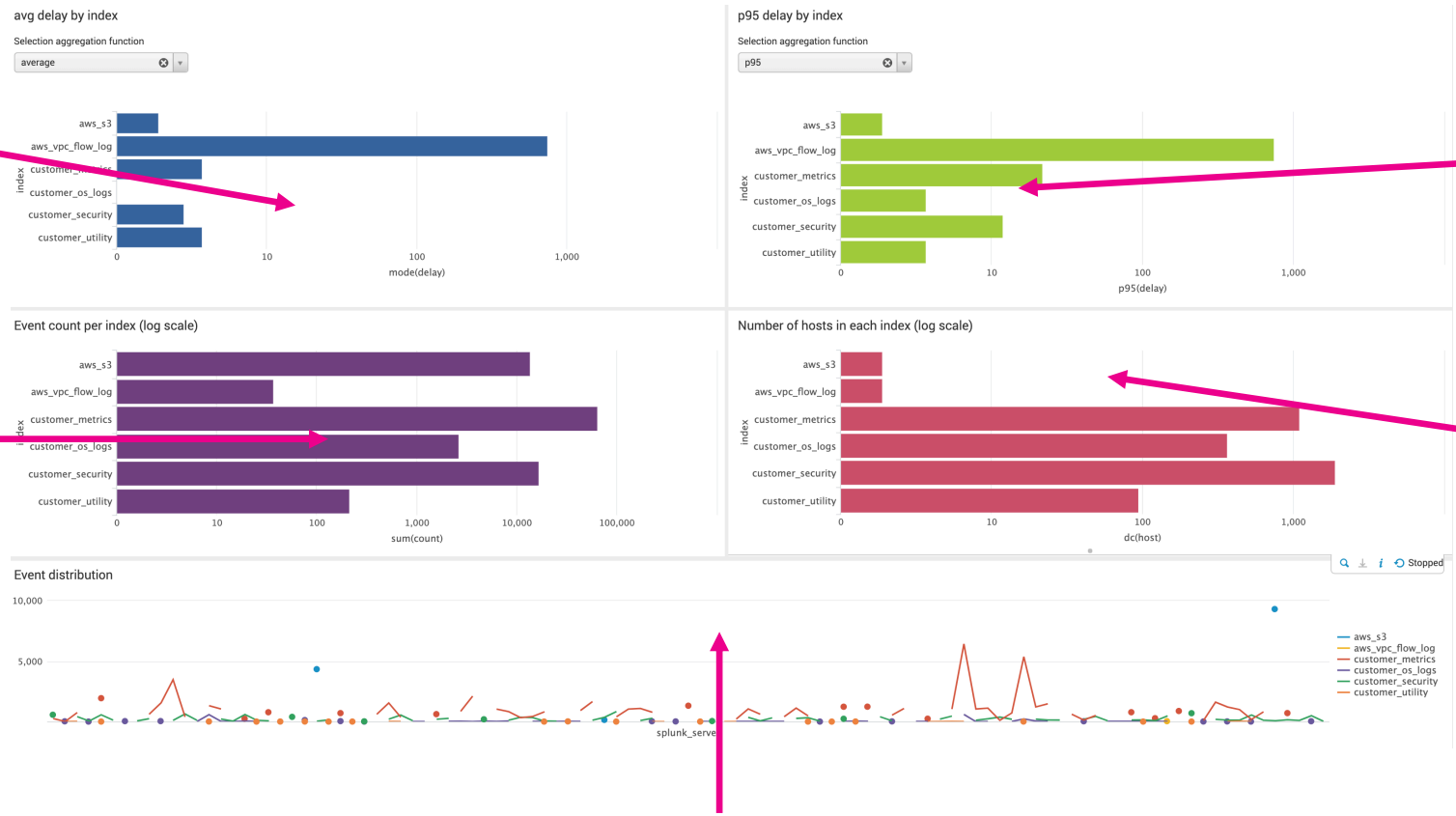
By default the results are all for the last second

The average delay per index

The maximum delay per index

The number of events received per index

The number of hosts per index



How events are distributed across the cluster

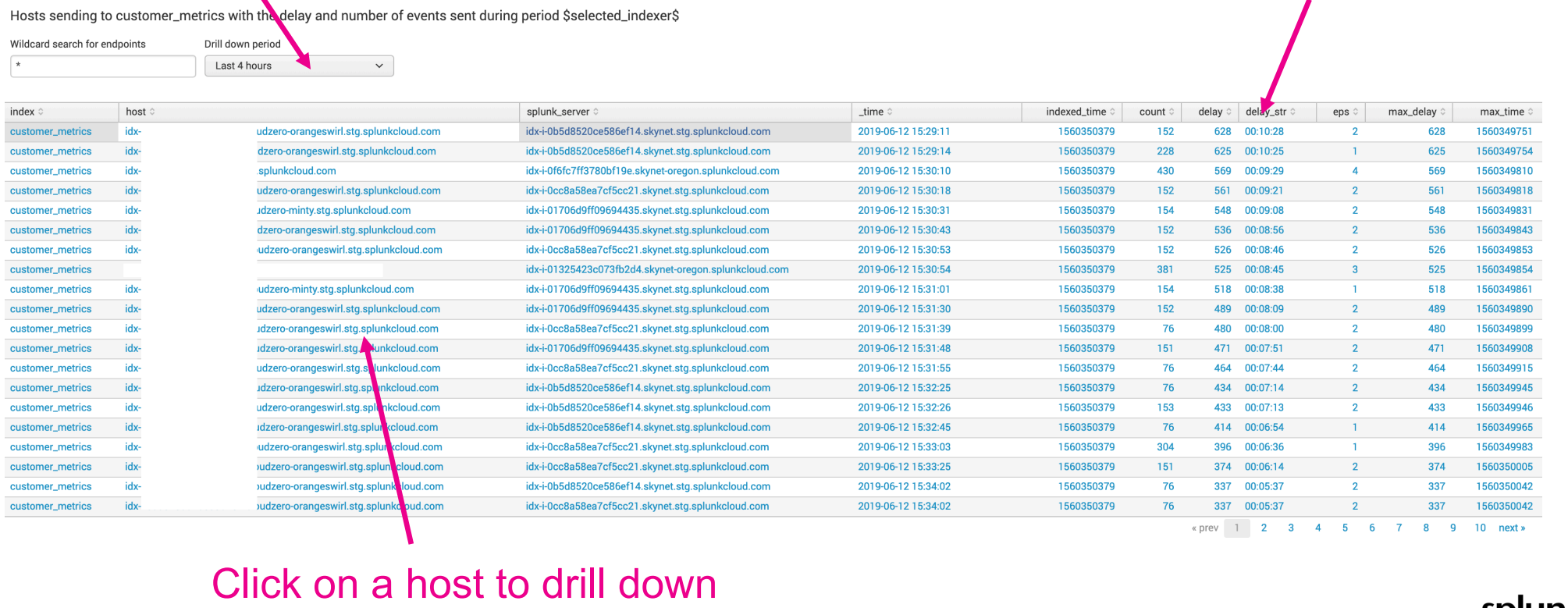
splunk> .conf19

Select an index by clicking on a series

Click on any chart to select an index

Select period for drill down

Sorted by descending delay



index	host	splunk_server	_time	indexed_time	count	delay	delay_str	eps	max_delay	max_time
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0b5d8520ce586ef14.skynet.stg.splunkcloud.com	2019-06-12 15:29:11	1560350379	152	628	00:10:28	2	628	1560349751
customer_metrics	idx- dzero-orangeswirl.stg.splunkcloud.com	idx-i-0b5d8520ce586ef14.skynet.stg.splunkcloud.com	2019-06-12 15:29:14	1560350379	228	625	00:10:25	1	625	1560349754
customer_metrics	idx- splunkcloud.com	idx-i-0f6fc7ff3780bf19e.skynet-oregon.splunkcloud.com	2019-06-12 15:30:10	1560350379	430	569	00:09:29	4	569	1560349810
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2019-06-12 15:30:18	1560350379	152	561	00:09:21	2	561	1560349818
customer_metrics	idx- udzero-minty.stg.splunkcloud.com	idx-i-01706d9ff09694435.skynet.stg.splunkcloud.com	2019-06-12 15:30:31	1560350379	154	548	00:09:08	2	548	1560349831
customer_metrics	idx- dzero-orangeswirl.stg.splunkcloud.com	idx-i-01706d9ff09694435.skynet.stg.splunkcloud.com	2019-06-12 15:30:43	1560350379	152	536	00:08:56	2	536	1560349843
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2019-06-12 15:30:53	1560350379	152	526	00:08:46	2	526	1560349853
customer_metrics	idx-	idx-i-01325423c073fb2d4.skynet-oregon.splunkcloud.com	2019-06-12 15:30:54	1560350379	381	525	00:08:45	3	525	1560349854
customer_metrics	idx- udzero-minty.stg.splunkcloud.com	idx-i-01706d9ff09694435.skynet.stg.splunkcloud.com	2019-06-12 15:31:01	1560350379	154	518	00:08:38	1	518	1560349861
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-01706d9ff09694435.skynet.stg.splunkcloud.com	2019-06-12 15:31:30	1560350379	152	489	00:08:09	2	489	1560349890
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2019-06-12 15:31:39	1560350379	76	480	00:08:00	2	480	1560349899
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-01706d9ff09694435.skynet.stg.splunkcloud.com	2019-06-12 15:31:48	1560350379	151	471	00:07:51	2	471	1560349908
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2019-06-12 15:31:55	1560350379	76	464	00:07:44	2	464	1560349915
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0b5d8520ce586ef14.skynet.stg.splunkcloud.com	2019-06-12 15:32:25	1560350379	76	434	00:07:14	2	434	1560349945
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0b5d8520ce586ef14.skynet.stg.splunkcloud.com	2019-06-12 15:32:26	1560350379	153	433	00:07:13	2	433	1560349946
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0b5d8520ce586ef14.skynet.stg.splunkcloud.com	2019-06-12 15:32:45	1560350379	76	414	00:06:54	1	414	1560349965
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2019-06-12 15:33:03	1560350379	304	396	00:06:36	1	396	1560349983
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2019-06-12 15:33:25	1560350379	151	374	00:06:14	2	374	1560350005
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0b5d8520ce586ef14.skynet.stg.splunkcloud.com	2019-06-12 15:34:02	1560350379	76	337	00:05:37	2	337	1560350042
customer_metrics	idx- udzero-orangeswirl.stg.splunkcloud.com	idx-i-0cc8a58ea7cf5cc21.skynet.stg.splunkcloud.com	2019-06-12 15:34:02	1560350379	76	337	00:05:37	2	337	1560350042

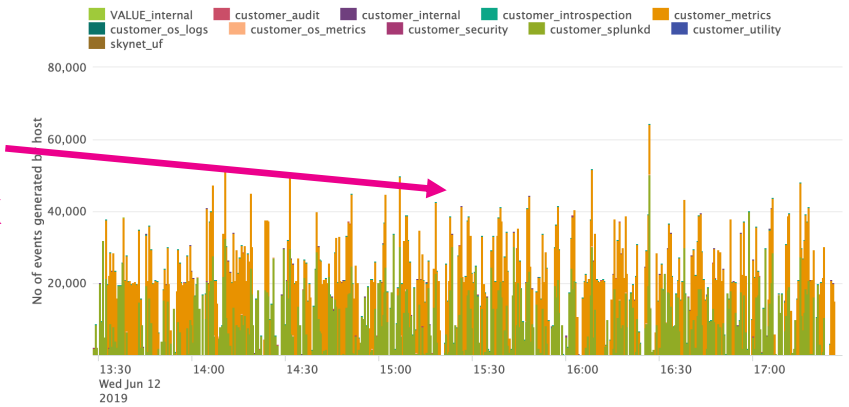
« prev 1 2 3 4 5 6 7 8 9 10 next »

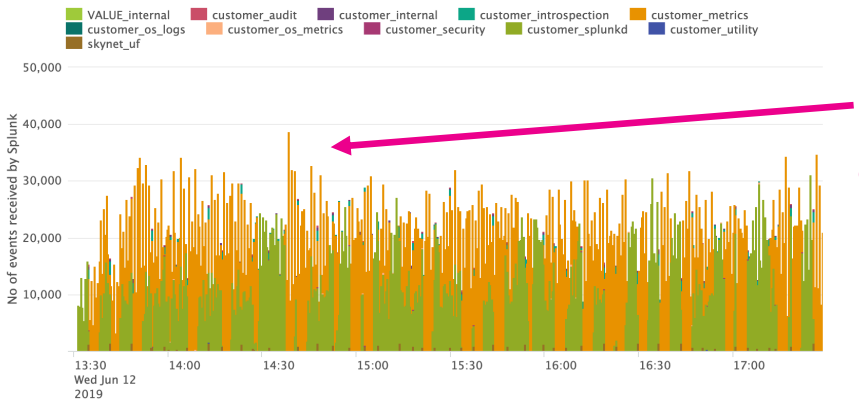
Click on a host to drill down

splunk> .conf19

Drill down to host to understand reasons for delay

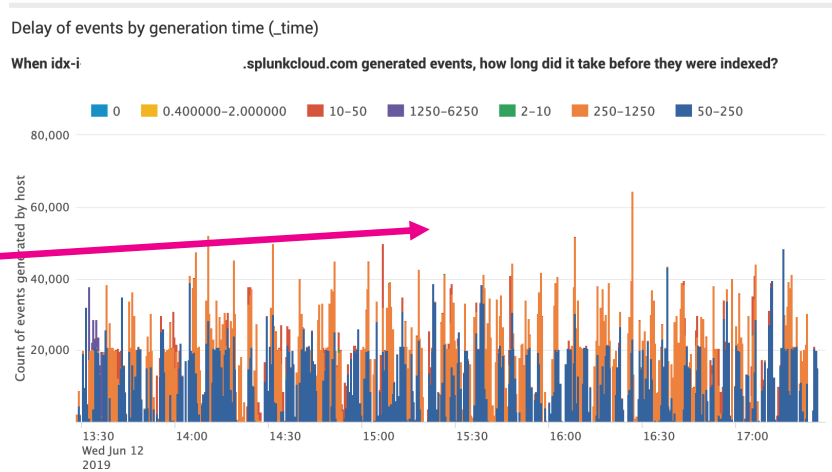
The number of events generated at time, per index

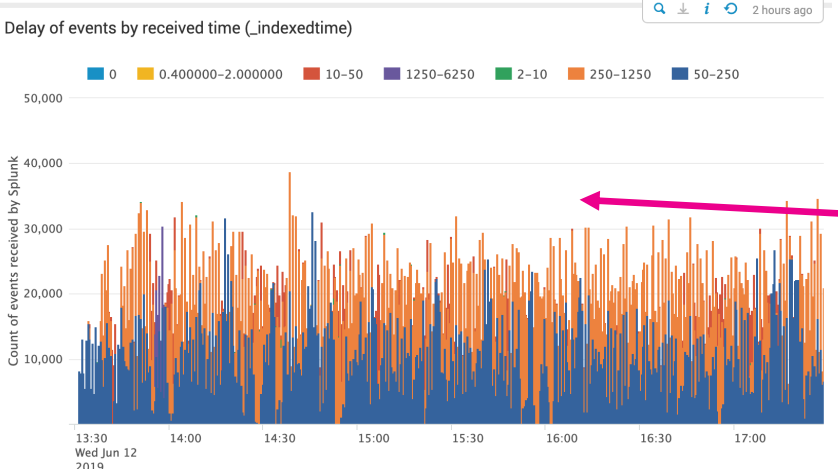




The number of events received at time, per index

The number of events generated at time, by delay





The number of events received at time, by delay

splunk> .conf19

The causes of event delay

Congestion

network latency, IO contention, pipeline congestion, CPU saturation, rate limiting

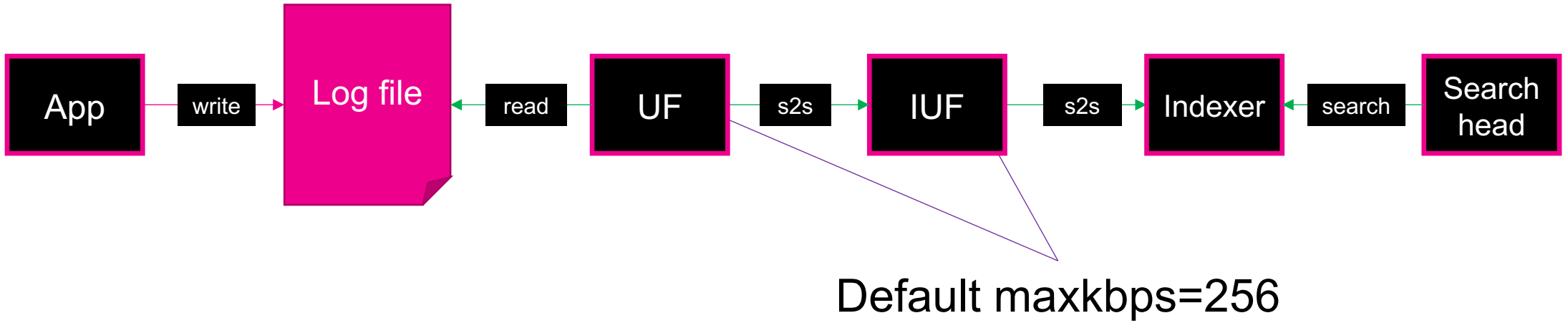
Clock skew

Time zones wrong, clock drift, parsing issue

Timeliness

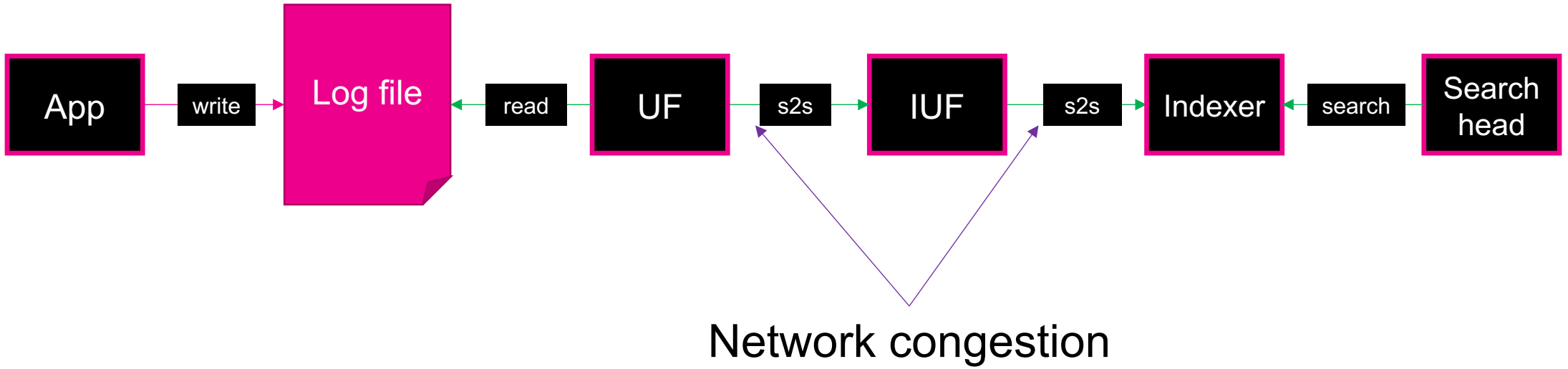
Historical load, polling APIs, scripted inputs, component restarts

Congestion: Rate limiting



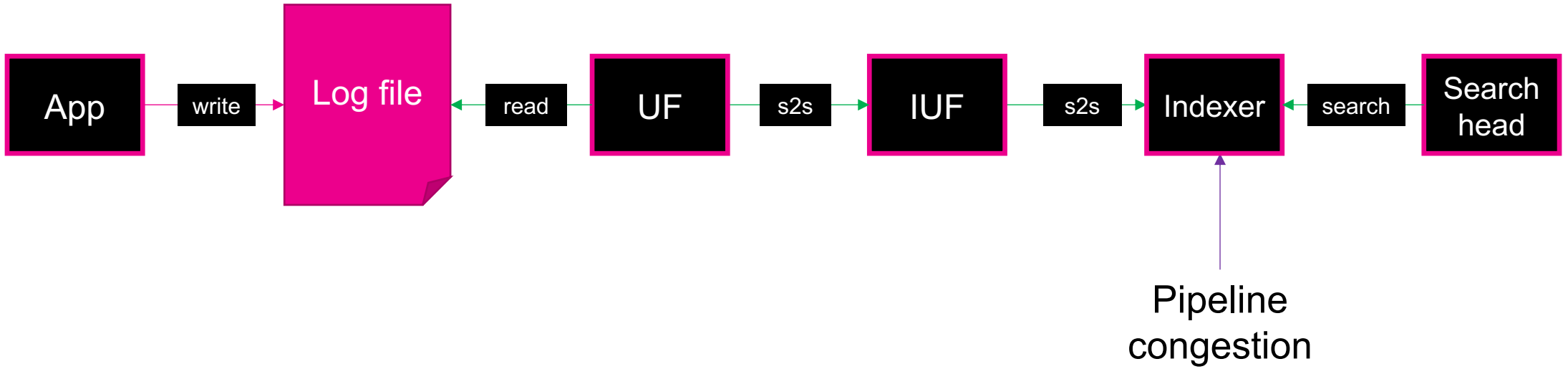
Rate limiting slows transmission and true event delay occurs

Congestion: Network



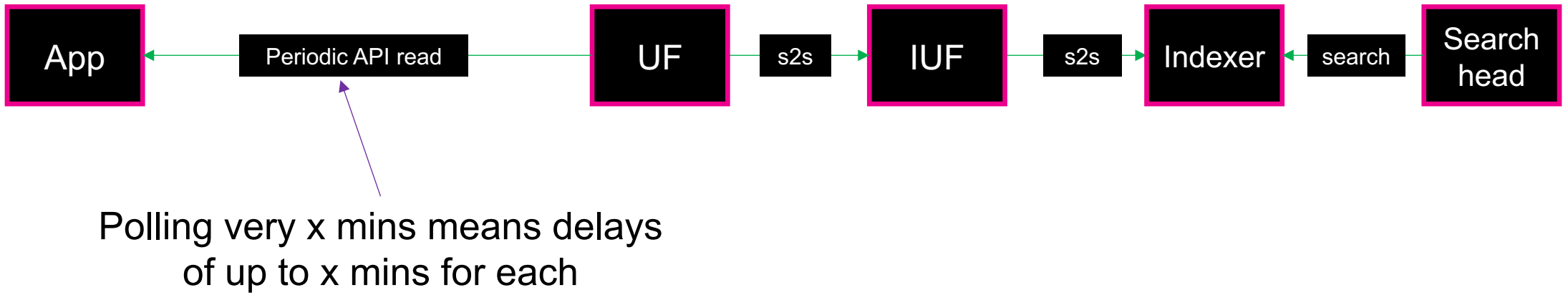
Network saturation acts like rate limiting and causes true event delay

Congestion: Indexer



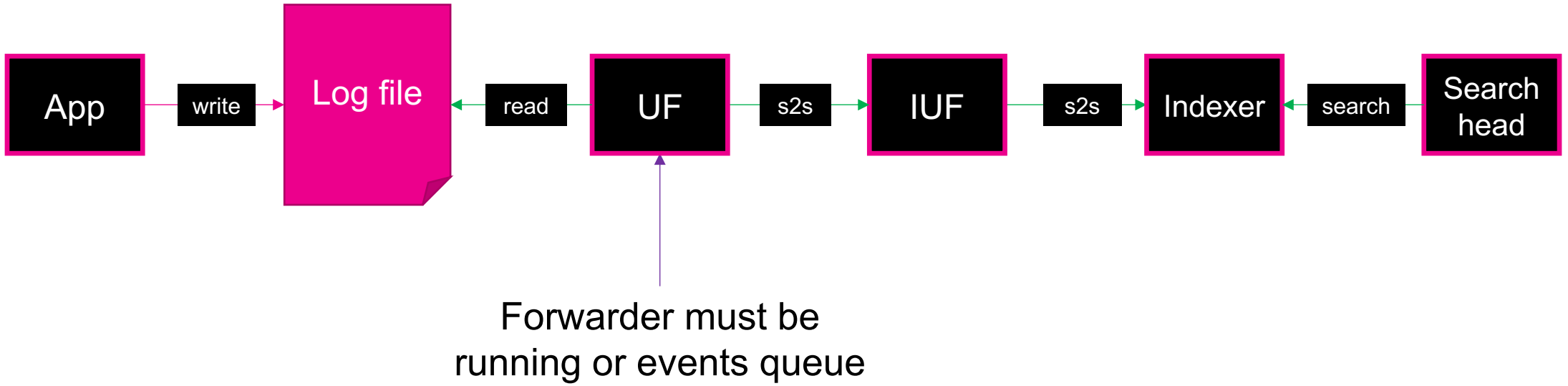
Excessive ingestion rates, FS IO problems, inefficient regex, inefficient line breaking cause all cause true event delay

Timeliness: Scripted inputs



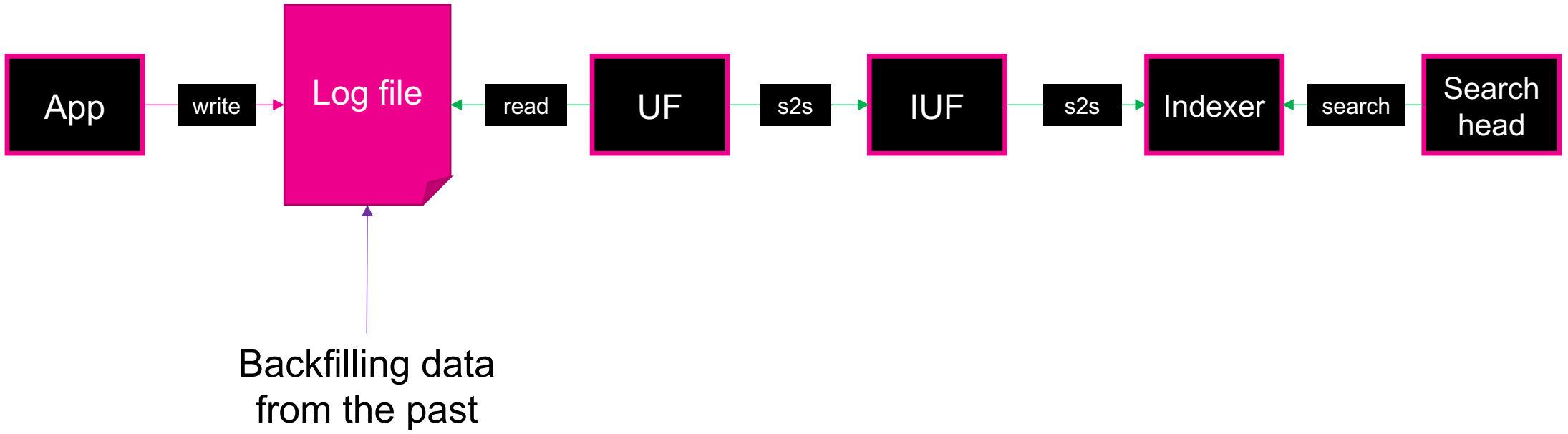
Increase polling frequency to reduce event delay

Timeliness: Offline components



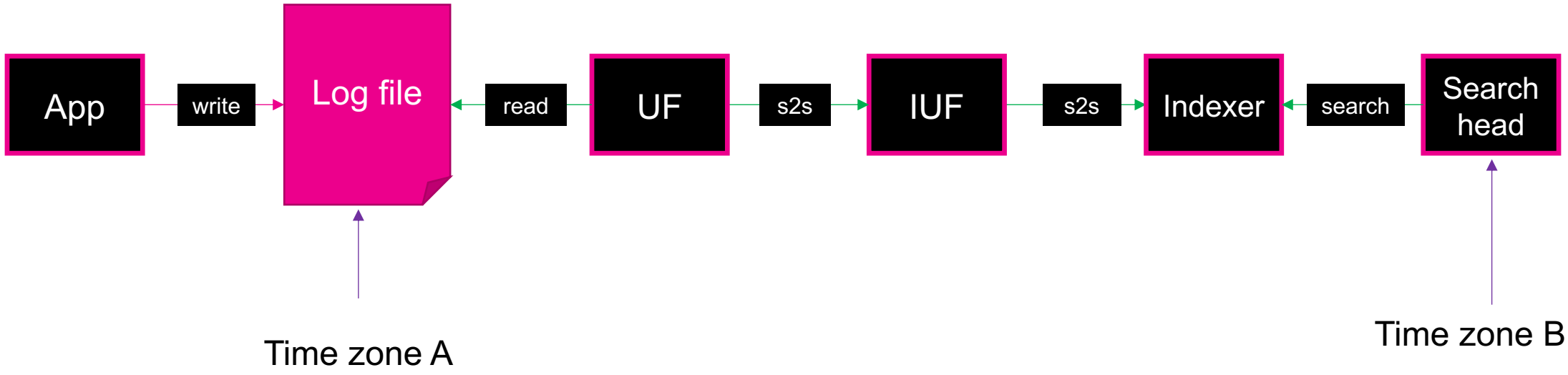
Restarting forwarders causes brief event delay

Timeliness: Historical data



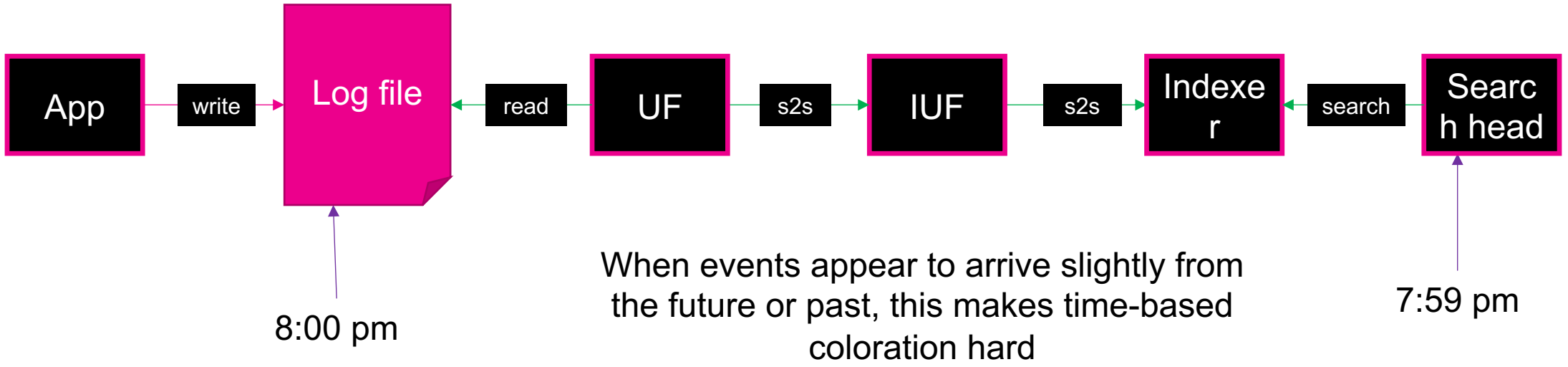
Loading historical data creates fake event delay

Clock Skew: Time zones



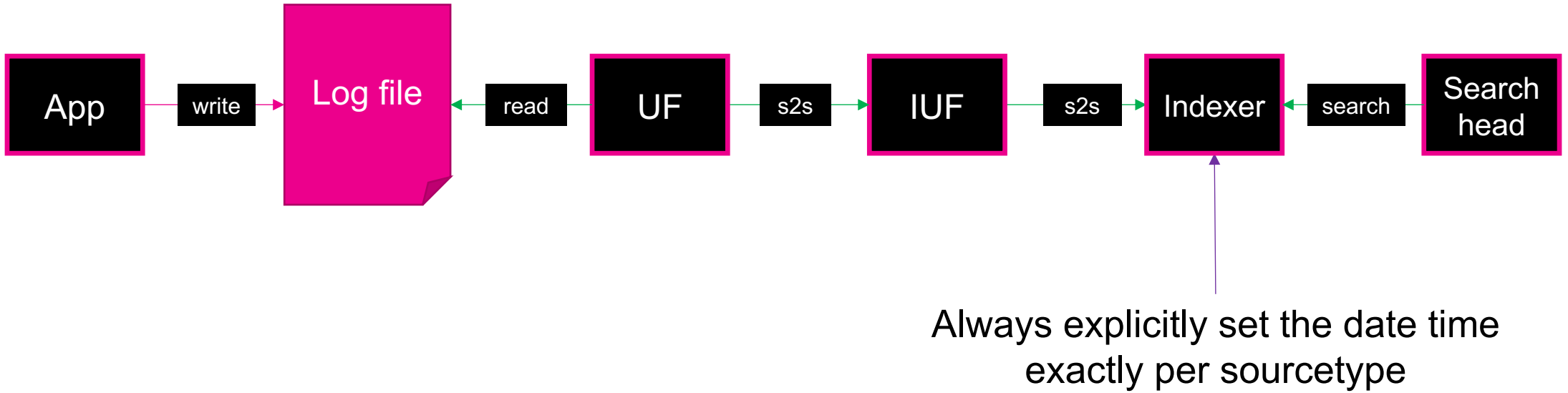
When time zones aren’t configured correctly event delay measurement is shifted into the past or future

Clock Skew: Drift



Use NTP to align all clocks across your estate to maximize the usefulness of Splunk

Clock Skew: Date time parsing problems



Automatic source typing assumes American date format when ambiguous

Reasons for poor event distribution

Sticky forwarders

Super giant forwarders

Badly configured intermediate forwarders

Indexer abandonment

Indexer starvation

Network connectivity problems

Single target

Forwarder bugs

TCP back off

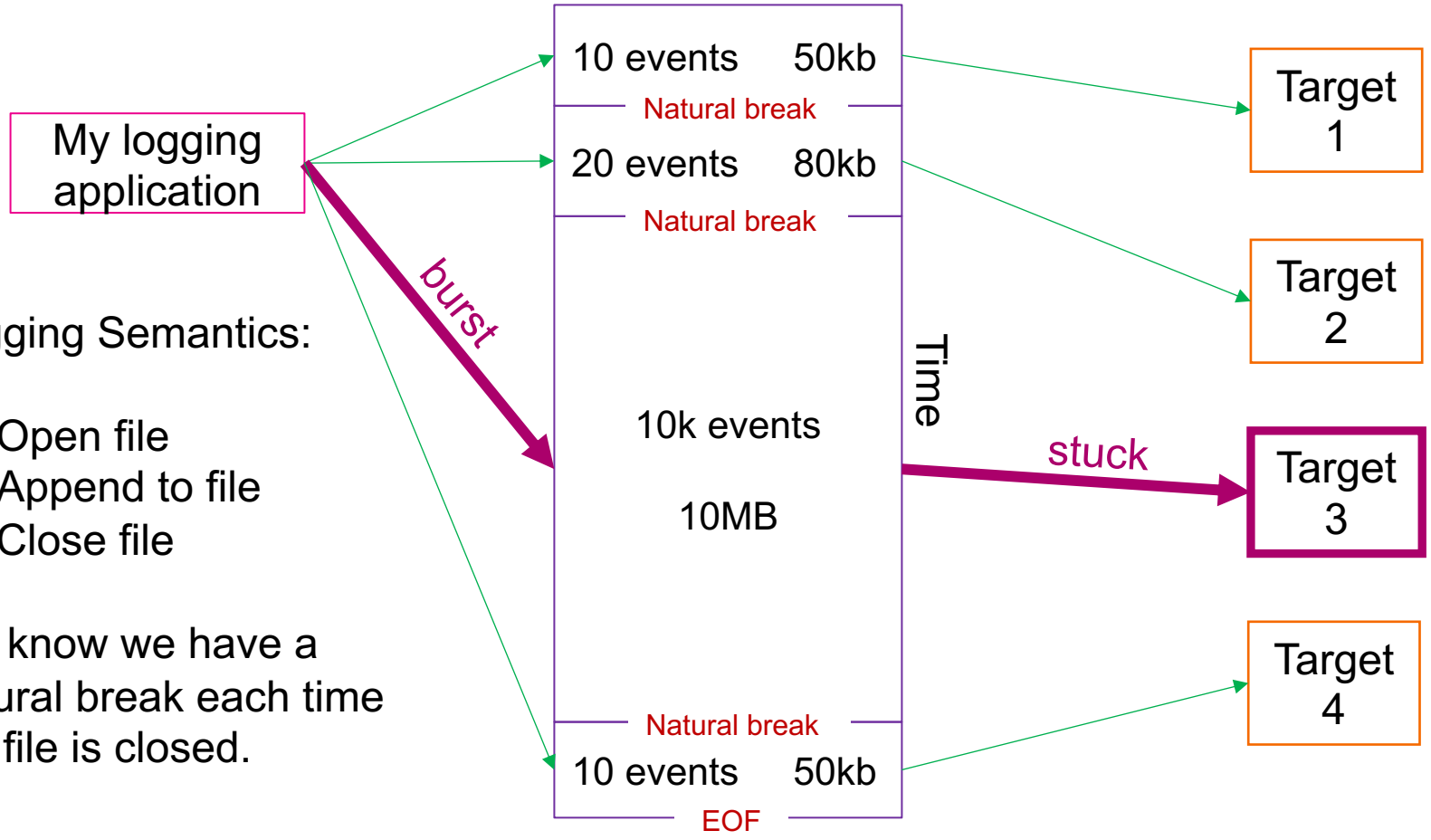
maxKBps

Channel saturation

HEC ingestion

Sticky Forwarders

The UF uses “natural breaks” to chunk up logs



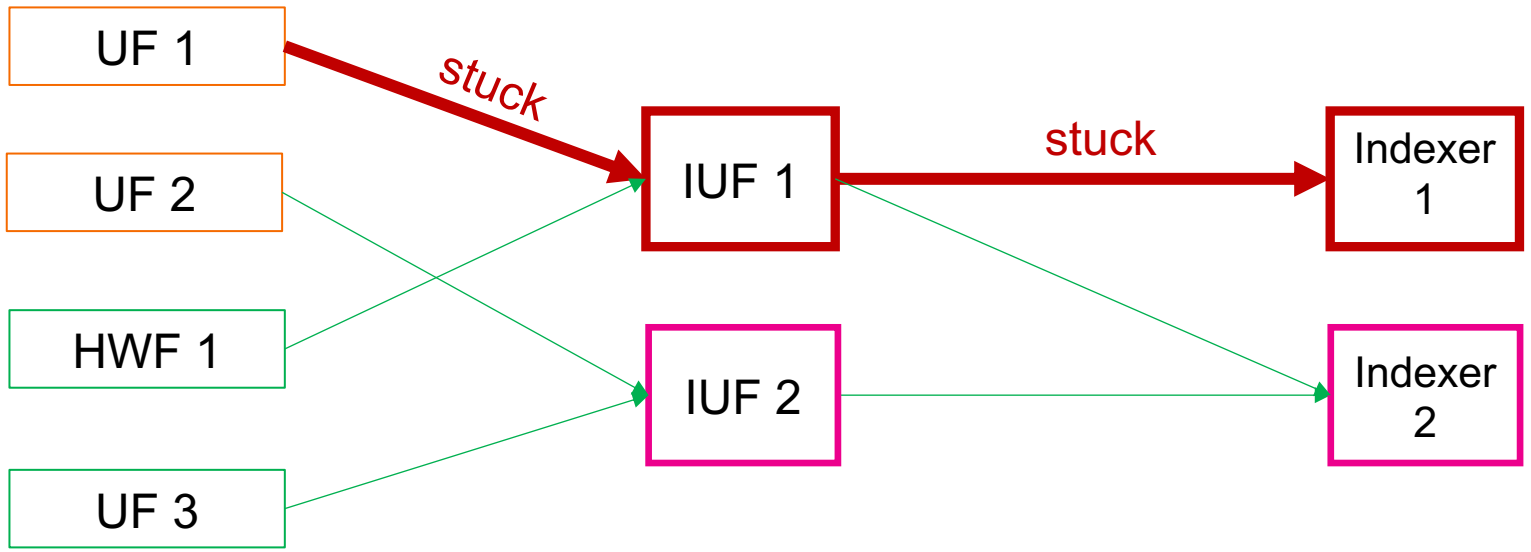
Logging Semantics:

1. Open file
2. Append to file
3. Close file

We know we have a natural break each time the file is closed.

When an application bursts the forwarder is forced to “stick” to the target until the application generates a natural break

The problem is exasperated with IUFs

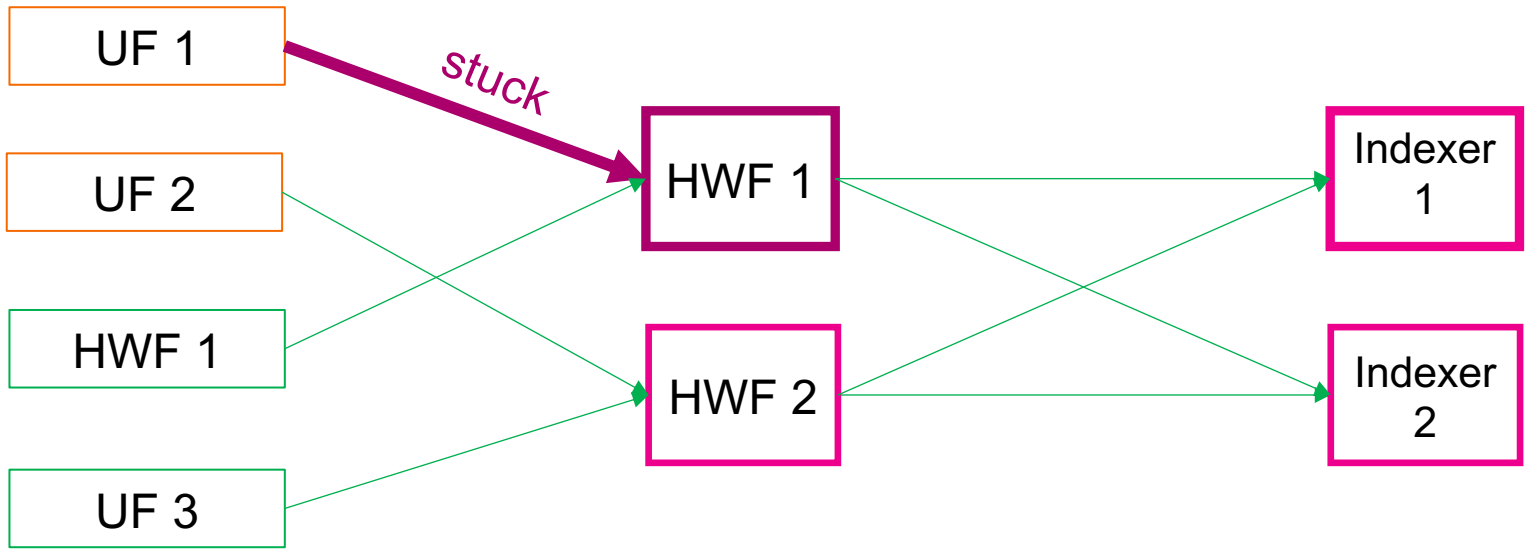


An intermediate universal forwarder (IUF) works with unparsed streams and can only switch away the incoming stream contains a break.

Connections can last for hours and this causes bad event distribution

IUFs cannot afford to be sticky and **must** switch on time

The problem doesn't exist with an intermediate HWF



The HWF parses data and forwarder events, not streams.

HWF can receive connections from sticky forwarders causing throughput problems

Heavy forwarders are generally considered a bad practise

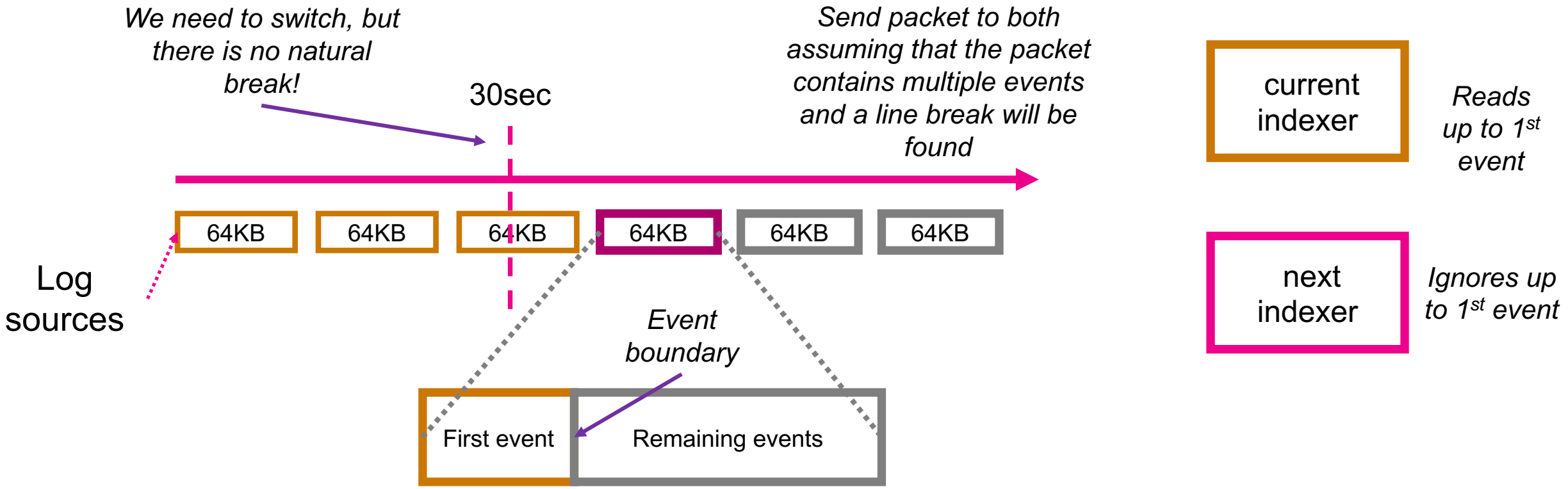
forceTimebasedAutoLB (outputs.conf)

```
forceTimebasedAutoLB = <boolean>
* Forces existing data streams to switch to a newly elected indexer every
  auto load balancing cycle.
* On universal forwarders, use the 'EVENT_BREAKER_ENABLE' and
  'EVENT_BREAKER' settings in props.conf rather than
'forceTimebasedAutoLB'
  for improved load balancing, line breaking, and distribution of events.
* Default: false
```

Forcing a UF to switch can create broken events and generate parsing errors

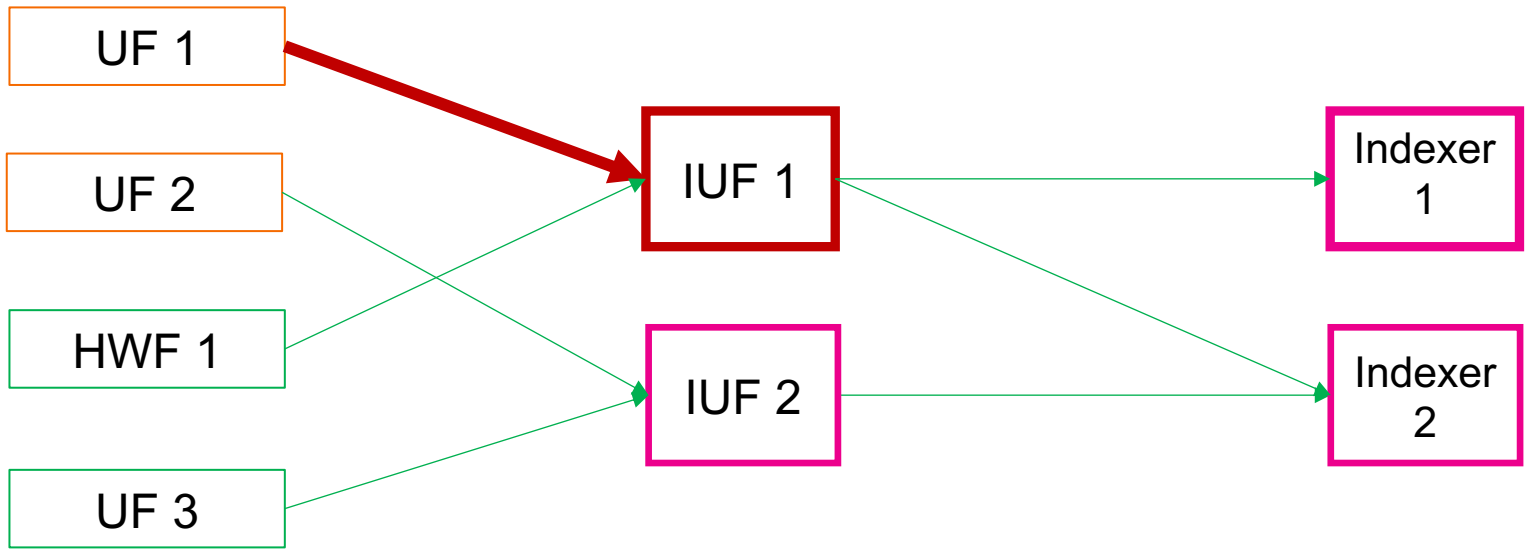
How forceTimeBasedAutoLB works

Splunk to Splunk protocol (s2s) uses datagrams of 64KB



Provided that an events doesn't succeed a s2s datagram the algorithm works perfectly

Applying forceTimeBasedAutoLB to IUF



The Intermediate Universal Forwarder will force switching without a natural break.

The indexers no longer get sticky sessions!

The intermediate forwarders still recieve sticky sessions

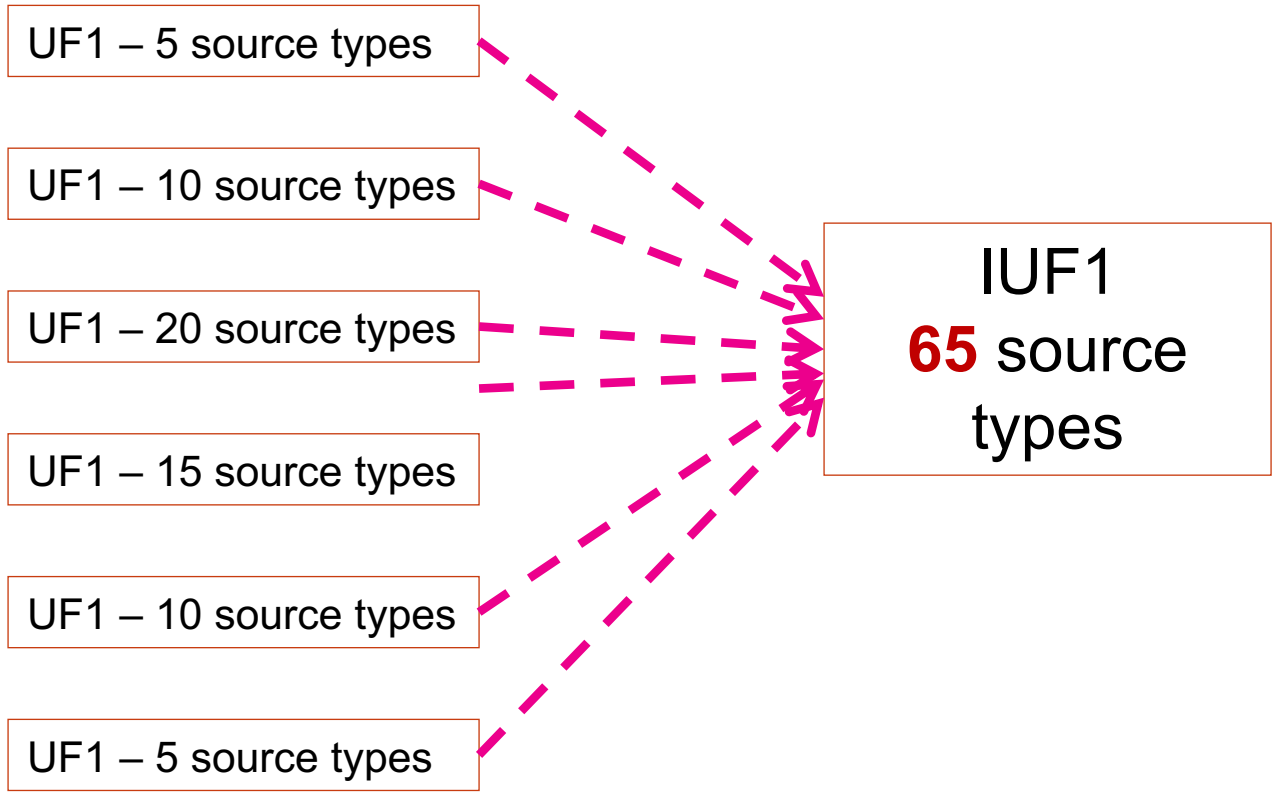
EVENT_BREAKER (outputs.conf)

```
# Use the following settings to handle better load balancing from UF.
# Please note the EVENT_BREAKER properties are applicable for Splunk Universal
# Forwarder instances only.

EVENT_BREAKER_ENABLE = [true|false]
* When set to true, Splunk software will split incoming data with a
  light-weight chunked line breaking processor so that data is distributed
  fairly evenly amongst multiple indexers. Use this setting on the UF to
  indicate that data should be split on event boundaries across indexers
  especially for large files.
* Defaults to false

# Use the following to define event boundaries for multi-line events
# For single-line events, the default settings should suffice

EVENT_BREAKER = <regular expression>
* When set, Splunk software will use the setting to define an event boundary at
  the end of the first matching group instance.
```

EVENT_BREAKER is configured with a regex **per sourcetype**

EVENT_BREAKER is complicated to maintain on IUF

Configure each sourcetype with EVENT_BREAKER so the forwarder doesn't get stuck the IUF.

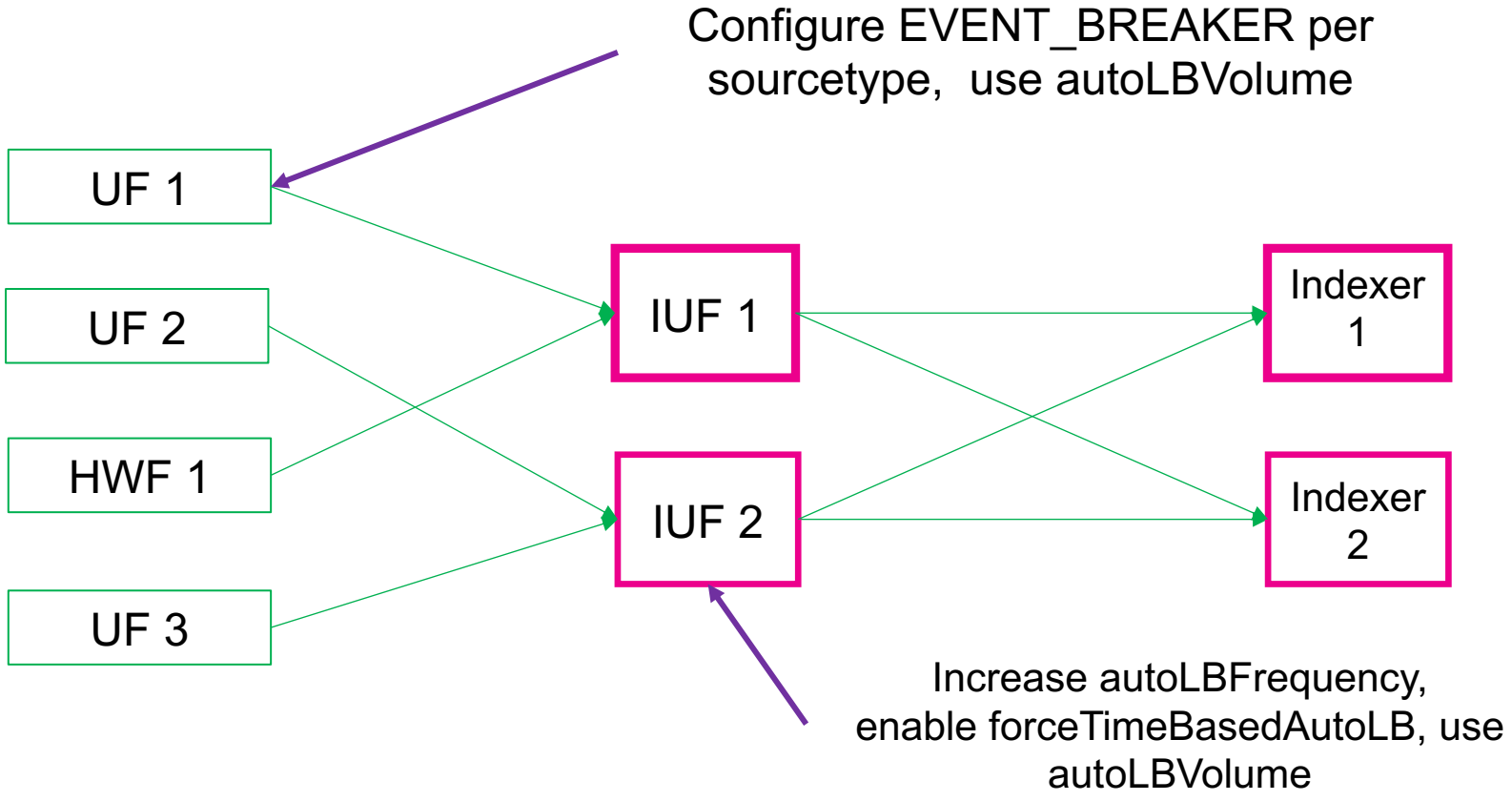


Trying to maintain EVENT_BREAKER on an IUF can be impractical due to aggregation of streams and source types.

If the endpoints all implement EVENT_BREAKER, it won't be triggered on the IUF

forceTimeBasedAutoLB is universal algorithm, EVENT_BREAKER is not

The final solution to sticky forwarders



Removal of sticky forwarders will lower event delay, improve event distribution, and improve search execution times

Configured correctly intermediate forwarders are great for improving event distribution.

Find all sticky forwarders by host name

```
index=_internal sourcetype=splunkd
      TERM(eventType=connect_done) OR
      TERM(eventType=connect_close)
| transaction
    startswith=eventType=connect_done
    endswith=eventType=connect_close
    sourceHost sourcePort host
| stats stdev(duration) median(duration) avg(duration) max(duration)
      by sourceHost
| sort - max(duration)
```

Intermediate LB will invalidate results

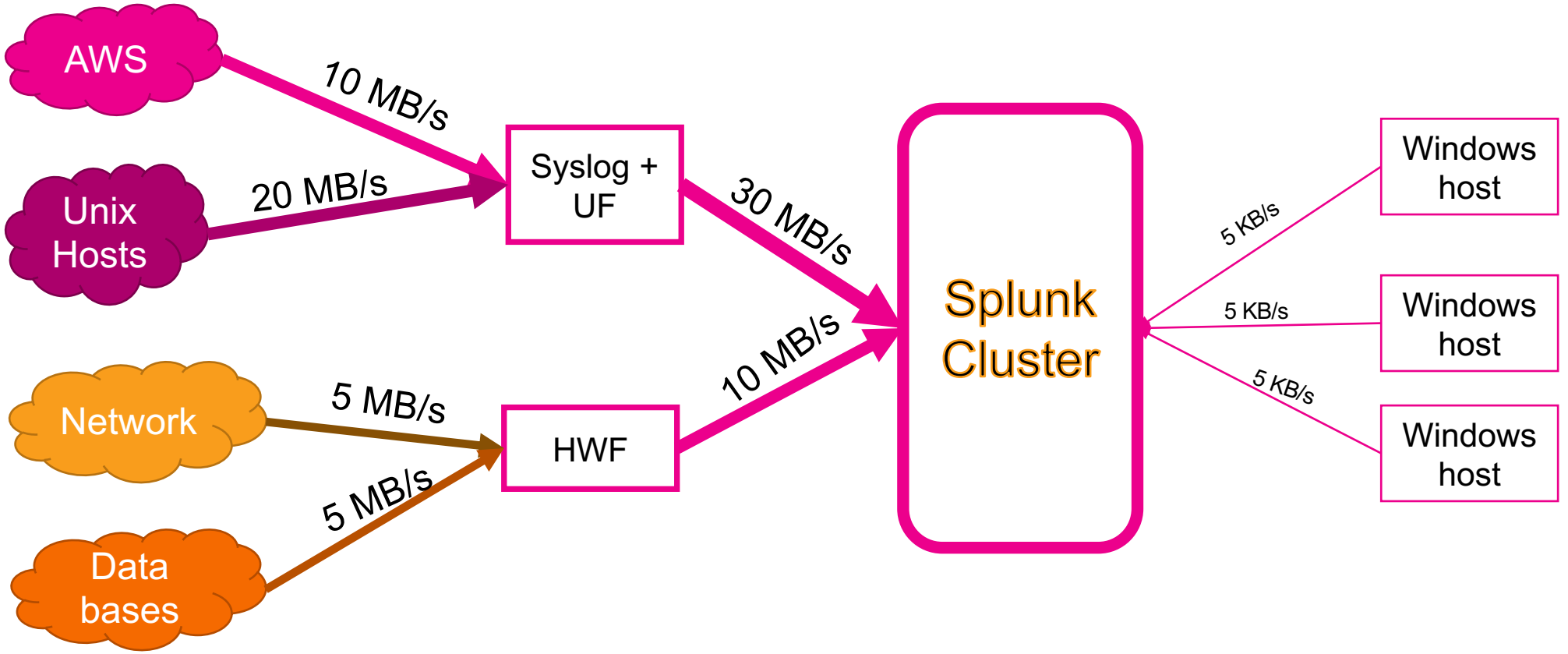
Find all sticky forwarders by hostname

```
index=_internal sourcetype=splunkd
      (TERM(eventType=connect_done) OR
      TERM(eventType=connect_close) OR
      TERM(group=tcpin_connections))
| transaction
      startswith=eventType=connect_done
      endswith=eventType=connect_close
      sourceHost sourcePort host
| stats stdev(duration) median(duration) avg(duration) max(duration)
      by hostname
| sort - max(duration)
```

Error prone as it requires that hostname = host

Super Giant Forwarders a.k.a. “laser beams of death”

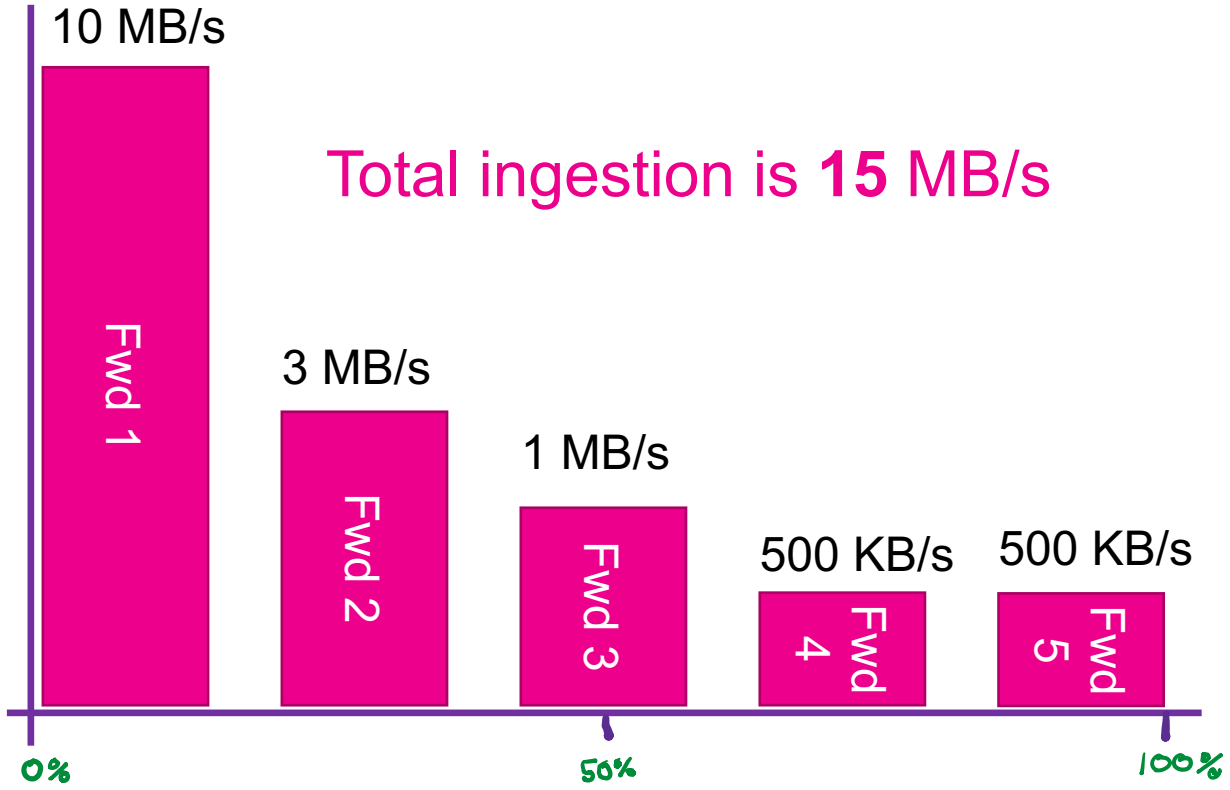
Not all Forwarders are born equal



Super giant forwarder make others look like rounding errors

splunk> .conf19

Understanding forwarder weight distribution

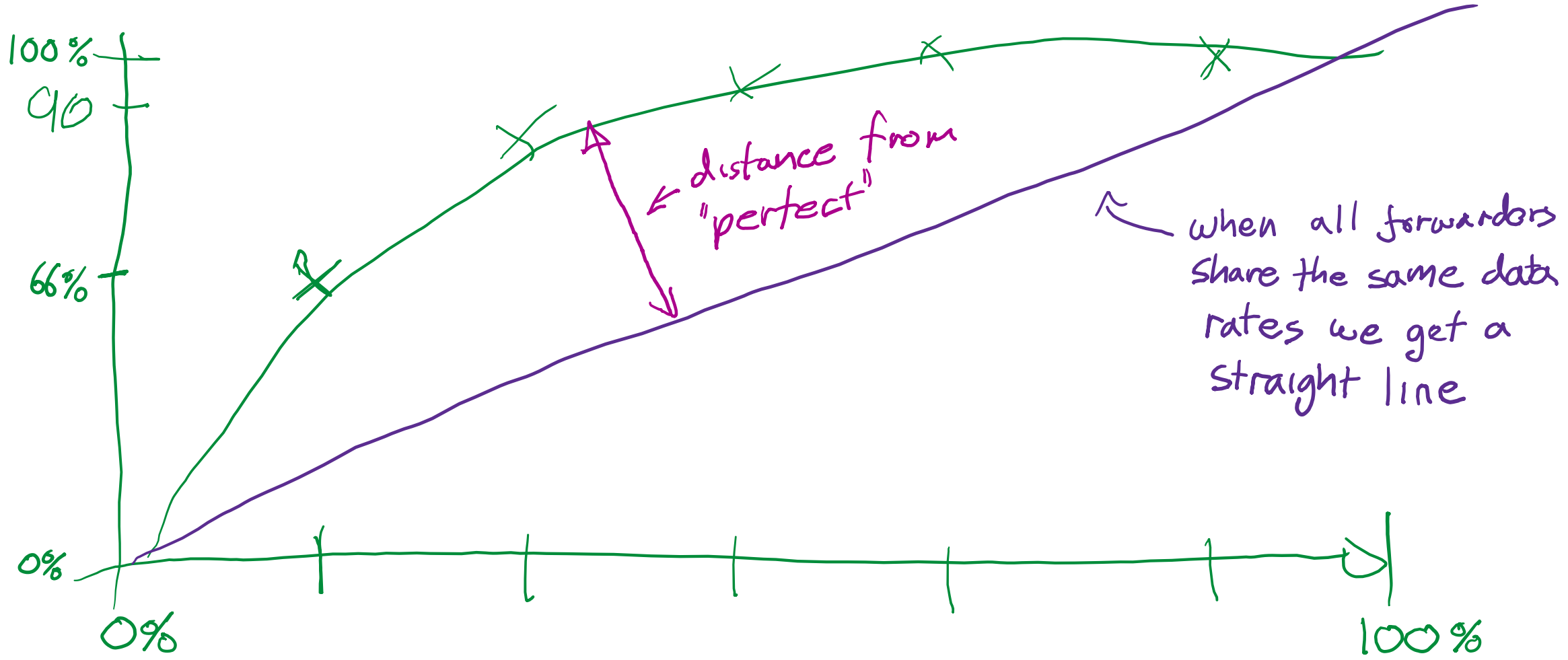


Reading from left to right

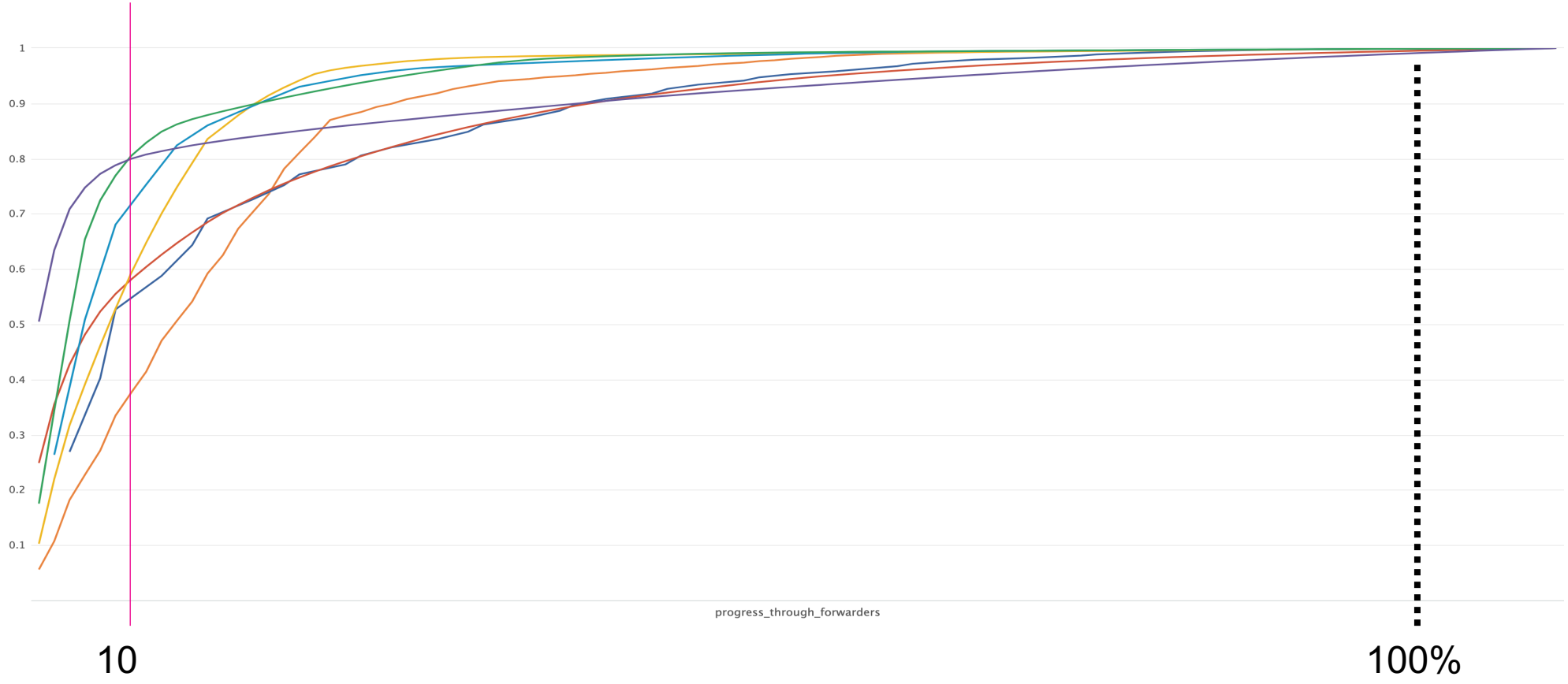
- 0% forwarders = 0% data
- 20% forwarders = 66% data
- 40% forwarders = 73% data
- 60% forwarders = 90% data
- 80% forwarders = 93% data
- 100% forwarders 100% data

We can plot these pairs as an chart to normalize and compare stacks.

Plotting normalized weight distribution



Examples of forward weight distribution



Data imbalance issues can be found on very large stack and must be addressed

Forwarder weight distribution search

```
index=_internal Metrics sourcetype=splunkd TERM(group=tcpin_connections) earliest=-4hr latest=now
    [| dbinspect index=_*
    | stats values(splunk_server) as indexer
    | eval search="host IN (".mvjoin(mvfilter(indexer!=""), ", ").")"]
| stats sum(kb) as throughput
    by hostname
| sort - throughput
| eventstats sum(throughput) as total_throughput
    dc(hostname) as all_forwarders
| streamstats sum(throughput) as accumlated_throughput count
    by all_forwarders
| eval coverage=accumlated_throughput/total_throughput,
    progress_through_forwarders=count/all_forwarders
| bin progress_through_forwarders bins=100
| stats max(coverage) as coverage
    by progress_through_forwarders all_forwarders
| fields progress_through_forwarders coverage
```

Find super giant forwarders and reconfigure

Super giant forwarders need careful configuration

1. Configure EVENT_BREAKER and / or forceTimeBasedAutoLB
2. Configure multiple pipelines (validate that they are being used)
3. Configure autoLBVolume and / or increase switching speed (keeping an eye on throughput)
4. Use INGEST_EVAL and random() to shard output data flows

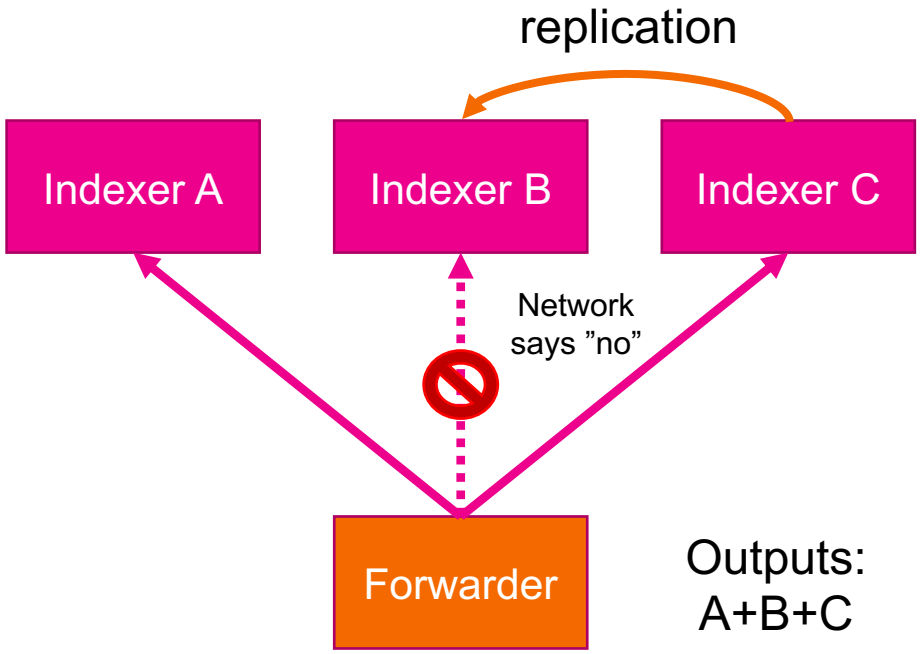
Indexer abandonment

Firewalls can block connections

A common cause for starvation is forwarders only able to connect to a subset of indexers due to network problems.

Normally a firewall or LB is blocking the connections

This is very common when the indexer cluster is increased in size.



Forwarders generate errors when they cannot connect

Find forwarders suffering network problems

```
index=_internal earliest=-24hrs latest=now sourcetype=splunkd
TERM(statusee=TcpOutputProcessor) TERM(eventType=*)
| stats count
    count(eval(eventType="connect_try")) as try
    count(eval(eventType="connect_fail")) as fail
    count(eval(eventType="connect_done")) as done
    by destHost destIp
| eval bad_output=if(try=failed,"yes","no")
```

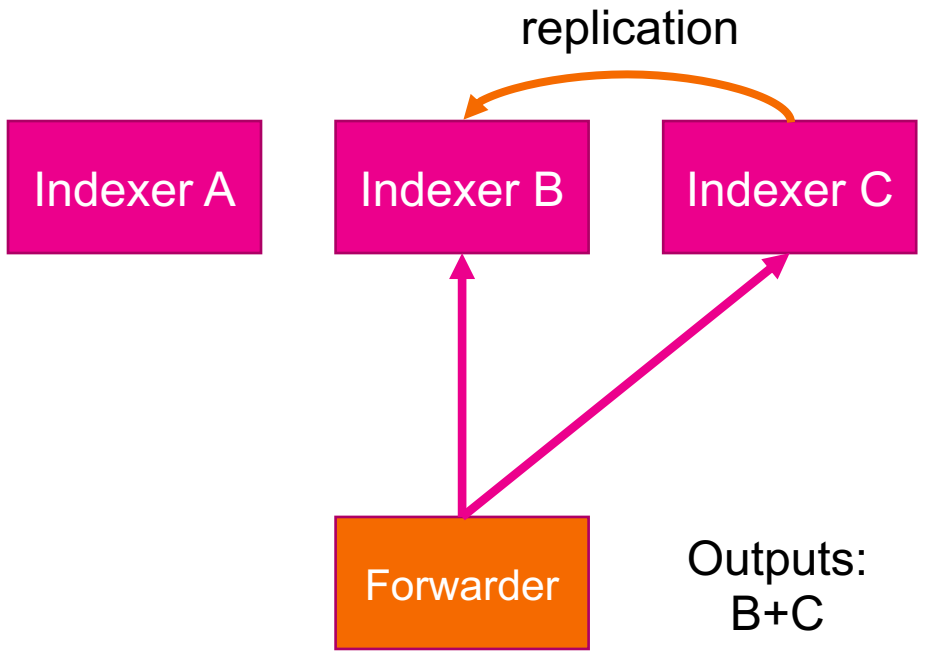
Search for forwarders logs to find those fail to connect to a target

Incomplete output lists create no errors

Forwarders can only send data to targets that are in their list.

This is very common when the indexer cluster is increased in size.

Encourage customers to use indexer discovery on forwarders so this never happens.



We must search for the absence of connections to find this problem

Do all indexers have the same forwarders?

```
index=_internal earliest=-24hrs latest=now sourcetype=splunkd TERM(eventType=connect_done)
TERM(group=tcpin_connections)
    [| dbinspect index=*
    | stats values(splunk_server) as indexer
    | eval host_count=mvcount(indexer),
        search="host IN (".mvjoin(mvfilter(indexer!=""), ", ").")"]
| stats count
    by host sourceHost sourceIp
| stats
    dc(host) as indexer_target_count
    values(host) as indexers_connected_to
    by sourceHost sourceIp
| eventstats max(indexer_target_count) as total_pool
| eval missing_indexer_count=total_pool-indexer_target_count
| where missing_indexer_count != 0
```

This search assumes that all indexers have the same forwarders. With multiple clusters and sites, this might not be true

Search for forwarders logs to find those fail to connect to a target

Site aware forwarder connections

```
index=_internal earliest=-24hrs latest=now sourcetype=splunkd TERM(eventType=connect_done) TERM(group=tcpin_connections)
    [| dbinspect index=*
    | stats values(splunk_server) as indexer
    | eval host_count=mvcount(indexer),
       search="host IN (".mvjoin(mvfilter(indexer!=""), ", ").")"]
| eval remote_forwarder=sourceHost." & ".if(sourceIp!=sourceHost," (".sourceIp.")","")
| stats count as total_connections
    by host remote_forwarder
| join host
    [ search index=_internal earliest=-1hrs latest=now sourcetype=splunkd CMMaster status=success site*
    | rex field=message max_match=64 "(?<site_pair>site\d+,\"[^\"]+)"
    | eval cluster_master=host
    | fields + site_pair cluster_master
    | fields - _*
    | dedup site_pair
    | mvexpand site_pair
    | dedup site_pair
    | rex field=site_pair "^(?<site_id>site\d+),\"(?<host>.*)"
    | eventstats count as site_size by site_id cluster_master
    | table site_id cluster_master host site_size]
| eval unique_site=site_id." & ".cluster_master
| chart
    values(site_size) as site_size
    values(host) as indexers_connected_to
    by remote_forwarder unique_site
| foreach site*
    [| eval "length_<<FIELD>>"=mvcount('<<FIELD>>')]
```

Sub-search returns indexer list

Sub-search computes indexer to site mapping

What sites forwarders connect to which site and what is the coverage for that site?

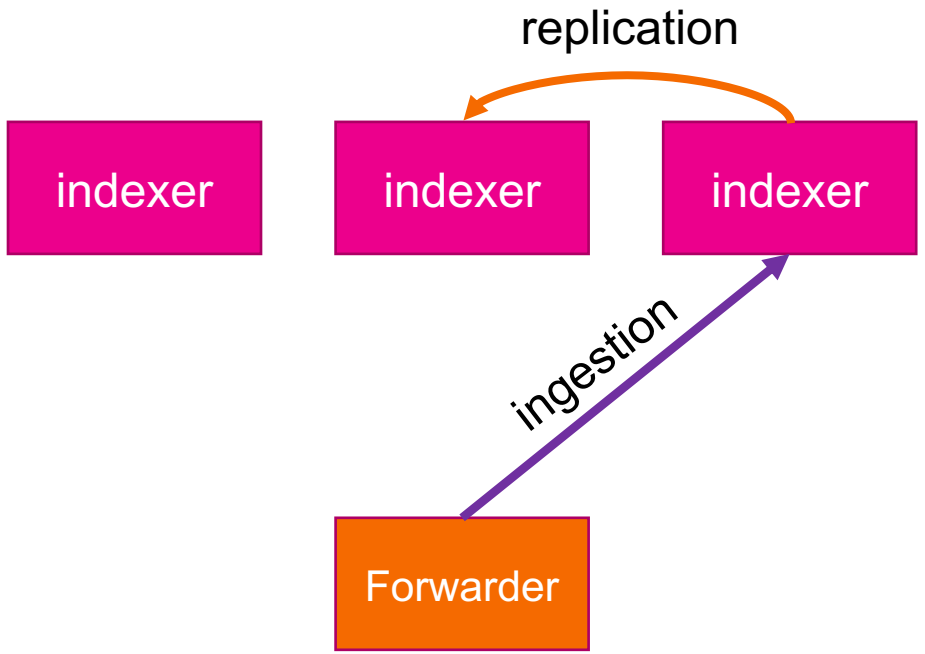
Indexer Starvation

What is indexer starvation?

When an indexer or an indexer pipeline has periods when it is starved of data.

It will continue to receive replicated data and be assigned primaries by the CM to search replicated cold buckets.

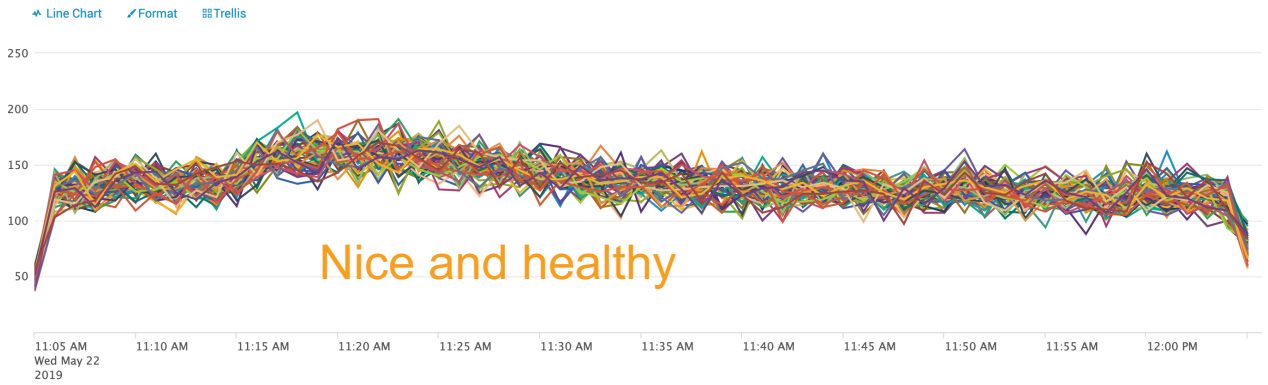
It will not search hot buckets without site affinity.



Indexers must receive data to participate in search

Smoke test: count connections to indexers

```
index=_internal earliest=-1hrs latest=now sourcetype=splunkd
TERM(eventType=connect_done) TERM(group=tcpin_connections)
    [| dbinspect index=*
    | stats values(splunk_server) as indexer
    | eval host_count=mvcount(indexer),
        search="host IN (".mvjoin(mvfilter(indexer!=""), ", ").")"]
| timechart limit=0 count minspan=31sec by host
```



This quick smoke test shows if the indexers in the cluster have obviously varying numbers of incoming connections.

When you see banding it either a smoking gun, or different forwarder groups per site.

Indexer starvation is guaranteed if there are not enough incoming connections.

Fix by increase connections by increasing switching frequency and the number of pipelines on forwarders

splunk> .conf19

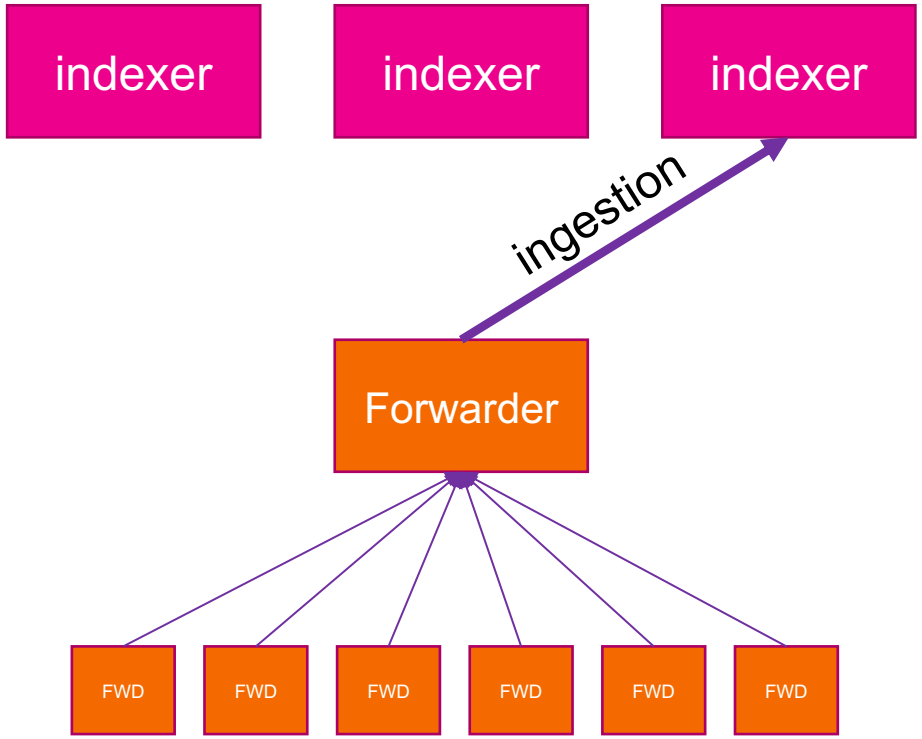
Funnel effect reduces connections

When an intermediate forwarder aggregates multiple streams, it creates a super giant forwarder.

You need to add more pipelines as each pipeline creates a new TCP output to the indexers.

Note that time division multiplexing doesn't mean 1 CPU = 1 pipeline, unless it is running at maximum rate of about 20 MB/s

Apply configuration with care.



How to instrument and tune IUFs

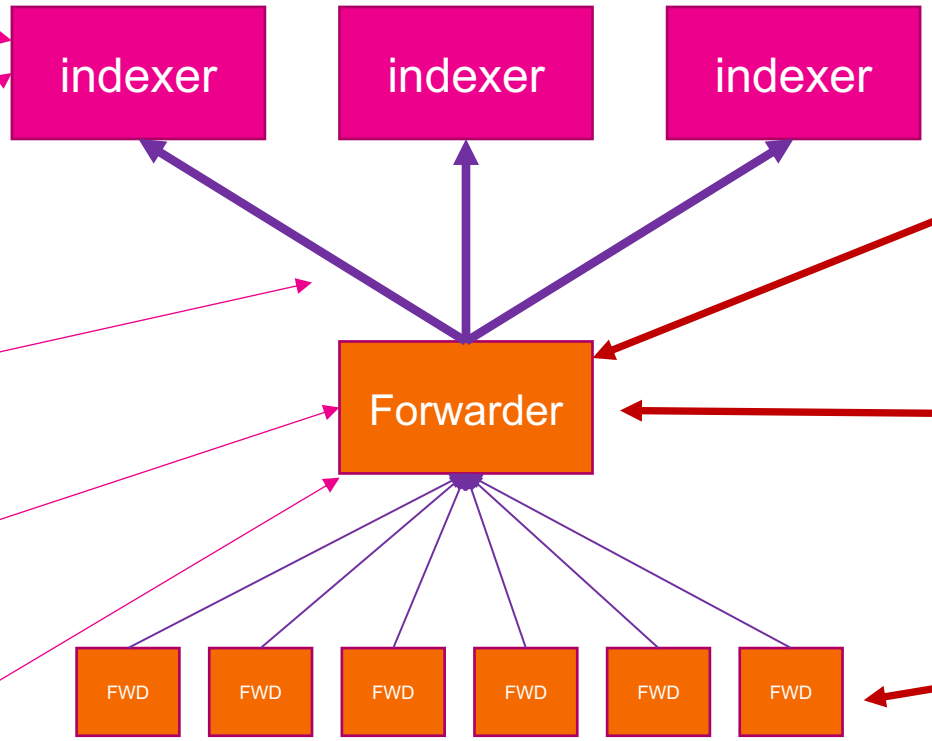
Monitor event delay to make sure it doesn't increase.

Monitor active channels to avoid s2s protocol churn

Monitor out going connections to understand cluster coverage

Monitor CPU, load average, pipeline utilization and pipeline blocking

Monitor incoming connections to ensure even distribution of workload over pipelines



Use autoLBVolume
Enable forceTimeBasedAutoLB
Lengthen timeBasedAutoLBFrequency

Add up to 16 pipelines but don't breach ~30% CPU

Find and reconfigure sticky forwarders to use EVENT_BREAKER

A well tuned intermediate forwarder can achieve 5% event distribution across 100 indexers within 5mins

.conf19

splunk>

Thank You!

Go to the .conf19 mobile app to

RATE THIS SESSION