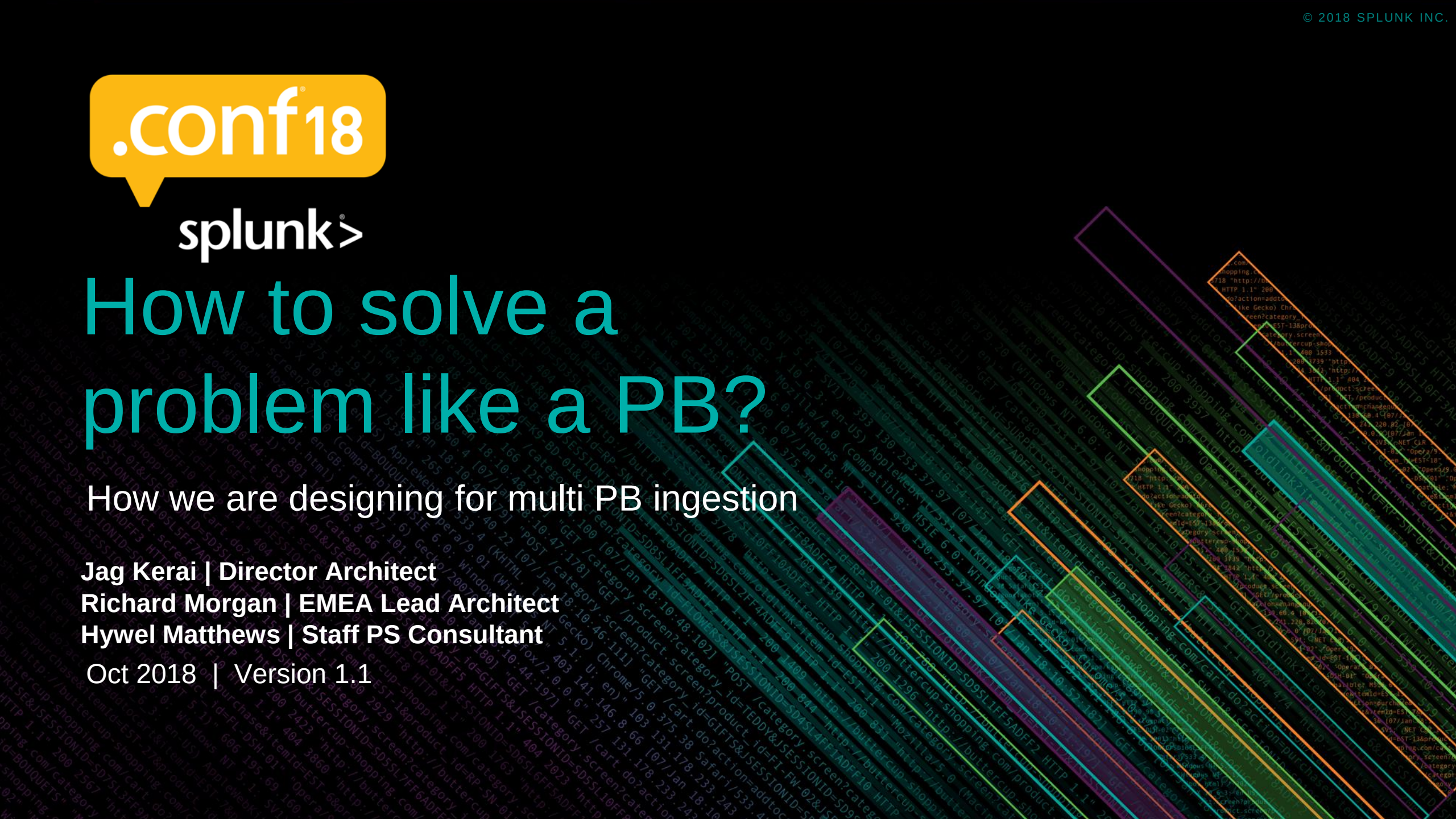


How to solve a problem like a PB?

How we are designing for multi PB ingestion

Jag Kerai | Director Architect
Richard Morgan | EMEA Lead Architect
Hywel Matthews | Staff PS Consultant

Oct 2018 | Version 1.1

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf18

RICHARD MORGAN

Lead Architect EMEA

Super Powers:
SPL, Splunk Architecture, Math

HYWEL MATTHEWS

Professional Services

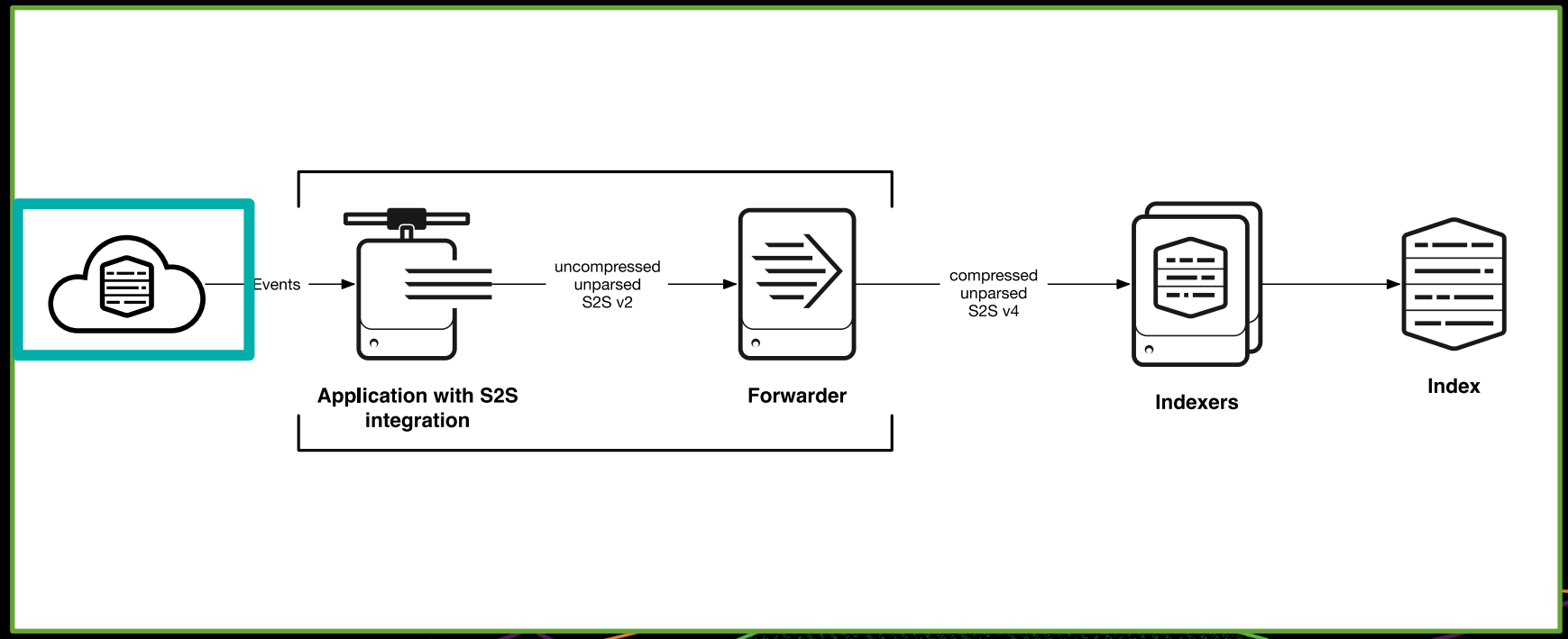
Super Powers:
Actual experience, detail, delivery, instructor

JAG KERAI

Director Architect

Super Powers:
All things splunkd, Splunk OG

"The realities of PB ingestion"



Ingestion rates

- 1 PB a day = 11.5 GB/s
- Data has seasonality +/- 50%
- Variable 5.7 GB/s – 17 GB/s
- 1 PB / 300 GB = 3300 indexers
- In a single pipeline Splunk can index about 20MB/s (1.7 TB a day)
- 17 GB/s / 20 MB/s = 850 indexers / pipelines

Search Space

- Searching a 5 minute window means parsing 5.1 TB of data
- Dense searches impractical without sampling
- Super sparse searches will work well
- Use very narrow time windows
- Lean on indexed fields where possible

Customer Challenge

- Service bus logging 2PB a day
- They need to debug transactions
- Store the data for 14 days
- Rack space is a premium
- Replace as engine for legacy platform

Transaction extraction use case

1. Within a 5 mins window …
2. Search for transaction parent
3. Return all sub-transaction ids
4. Search for sub-transactions + parent
5. Return results

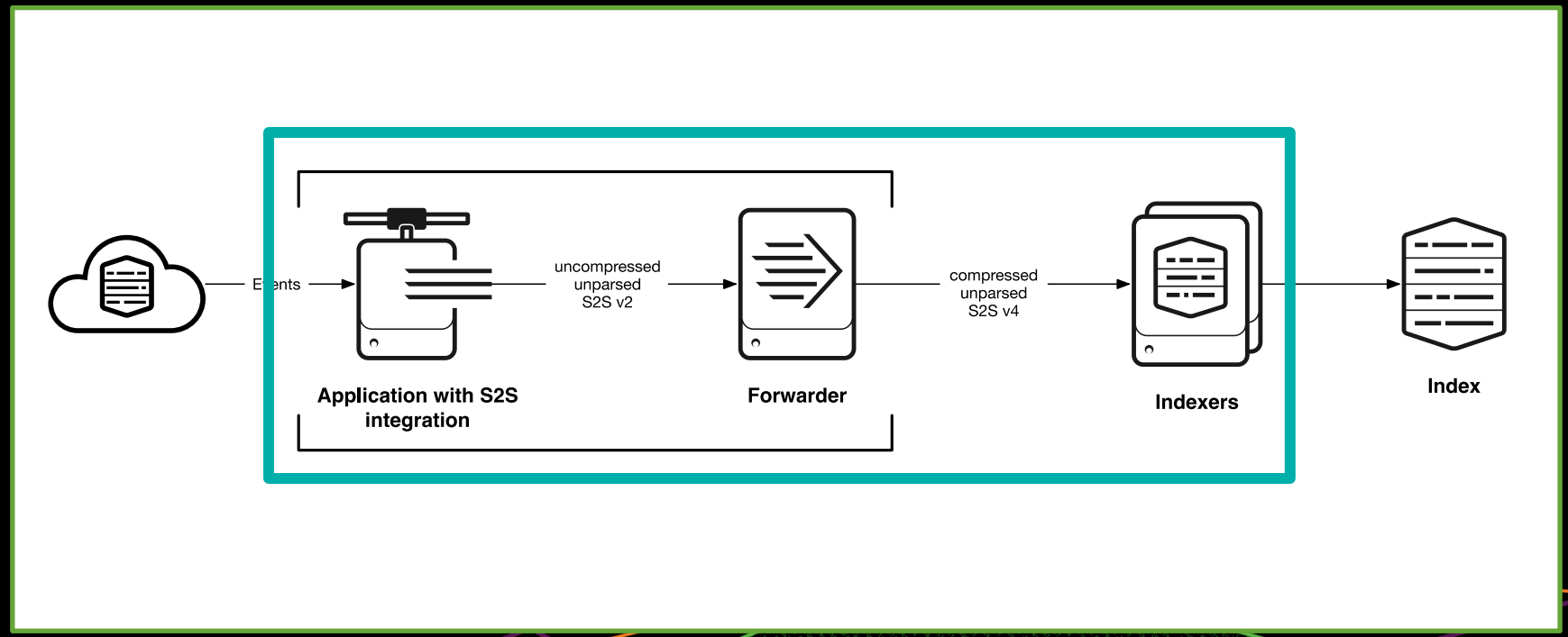
Perfect match for Splunk bloom filters!

1. Almost all searches small windows
2. Almost all searches super sparse
3. Design carefully to maximize search
4. Tune to increase ingestion efficiency
5. Buy platinum servers
6. Reduce infrastructure to minimum

POC Won!

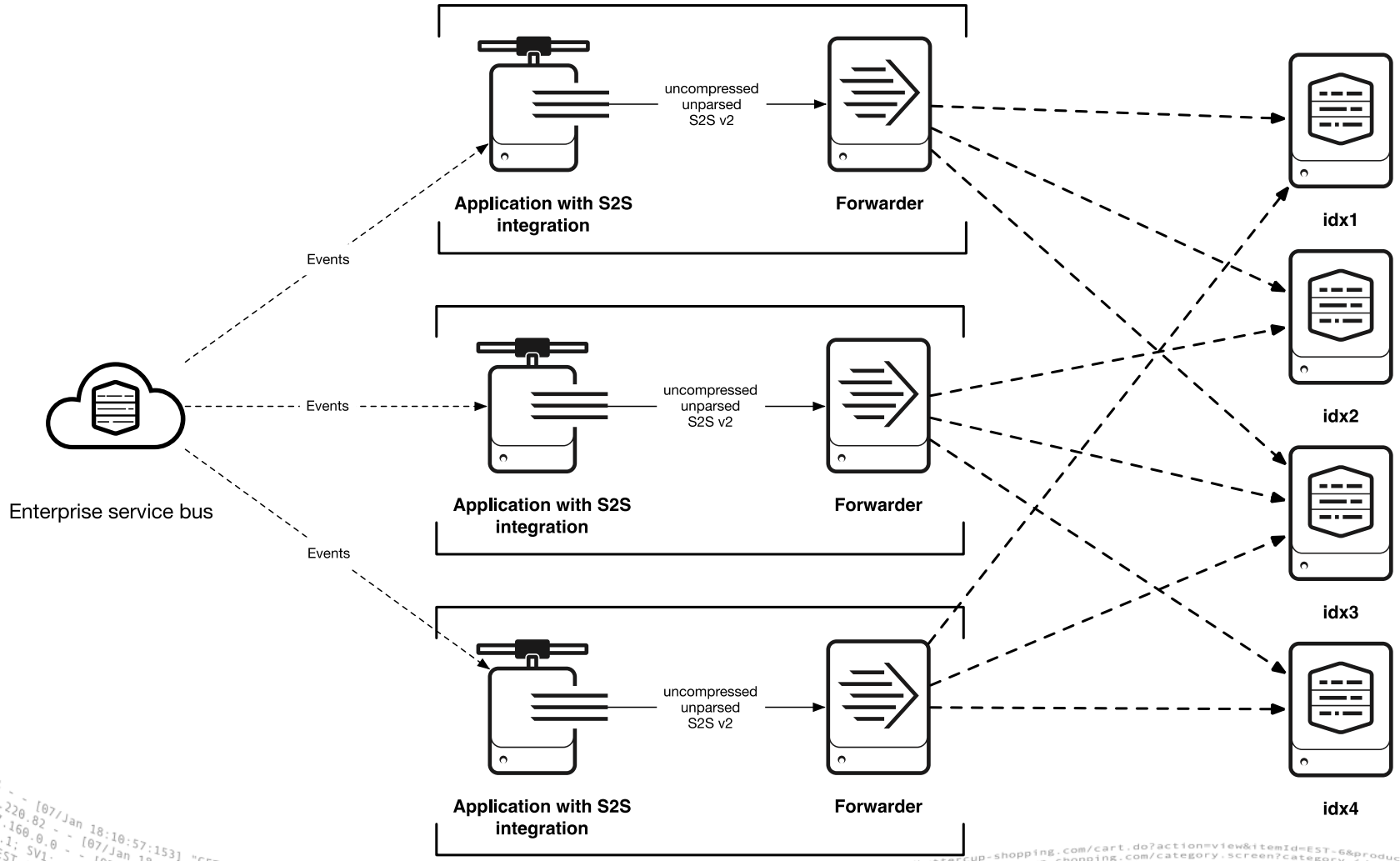
1. Achieved 3.5 TB an indexer on 7.0.1
2. 1 PB / 3.5 TB = 290 indexers
3. Doubled search load requirements
4. Resource utilization very low …

“High level Architecture”



Scale out Architecture

Things look normal at a high level



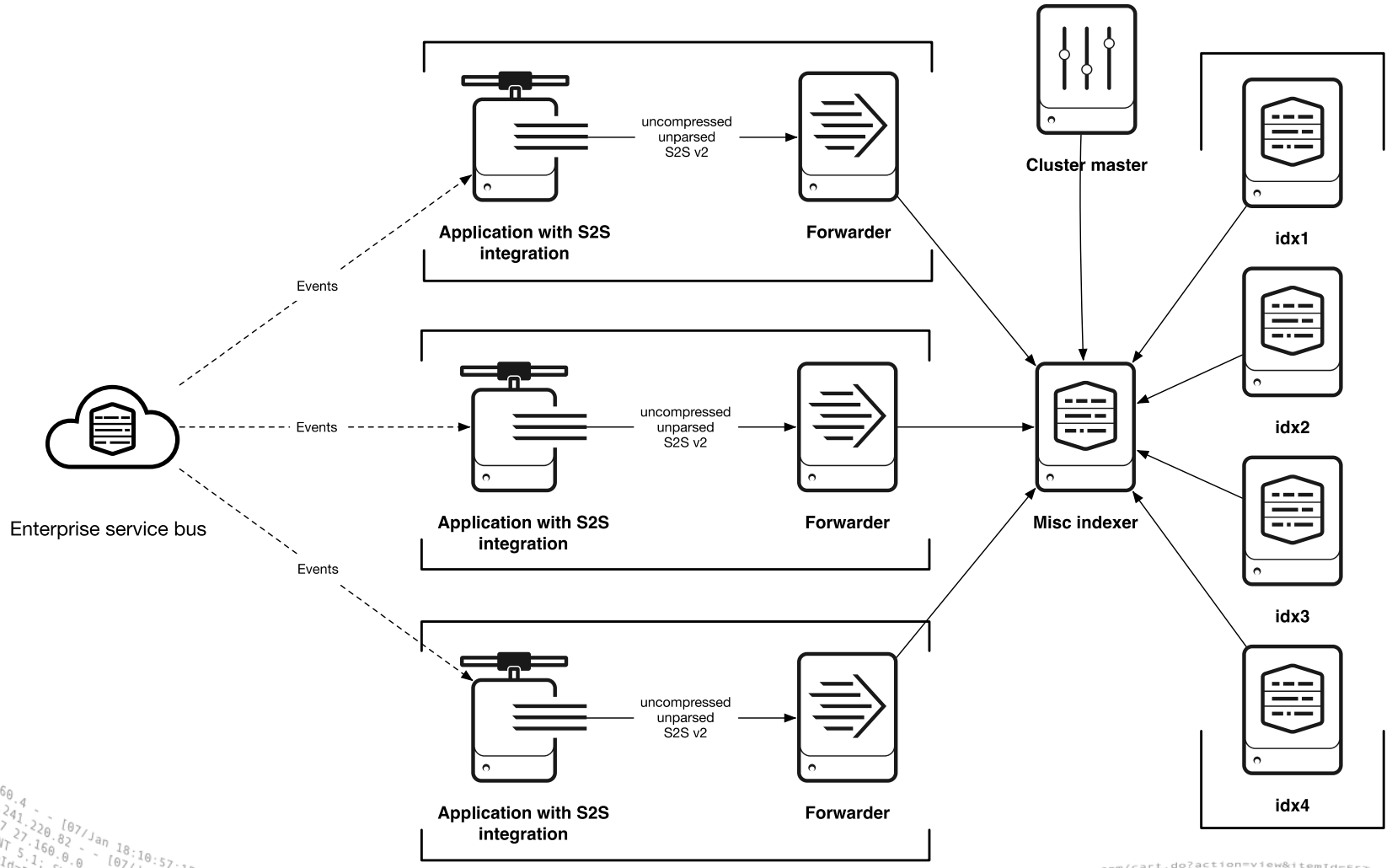
Application generating the events stream to a local UF and then are load balanced to the indexing tier.

This allows for scaling the indexing and the logging tier independently.

This separation of concerns makes the solution more tolerant to a stalling indexers

Removed complexity where possible (KISS)

Rerouted internal logging and disabled replication



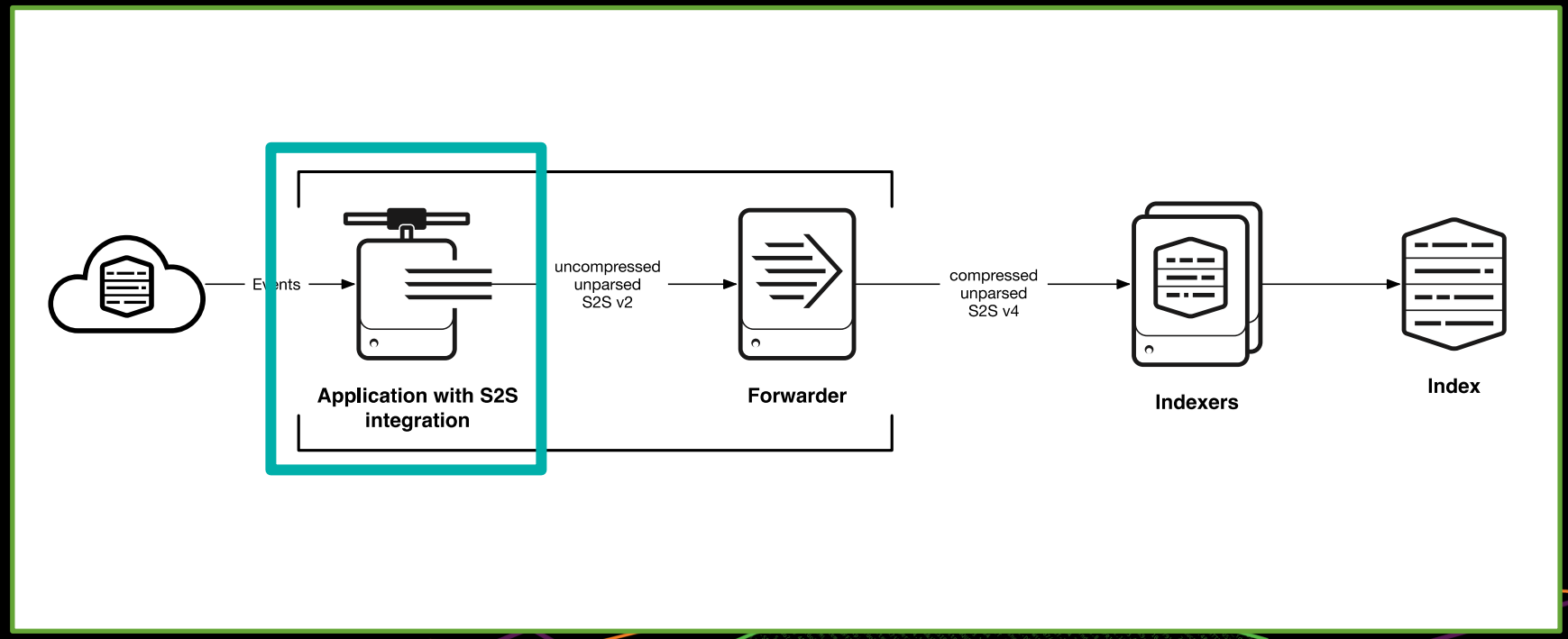
A cluster master exists for coordination of indexers and to run the DMC – but replication is disabled

A dedicated indexer is used for collecting the component logs, allowing for measurements to be taken without impact to the system.

Real time stats on ingestion!

splunk> .conf18

“S2S Application”



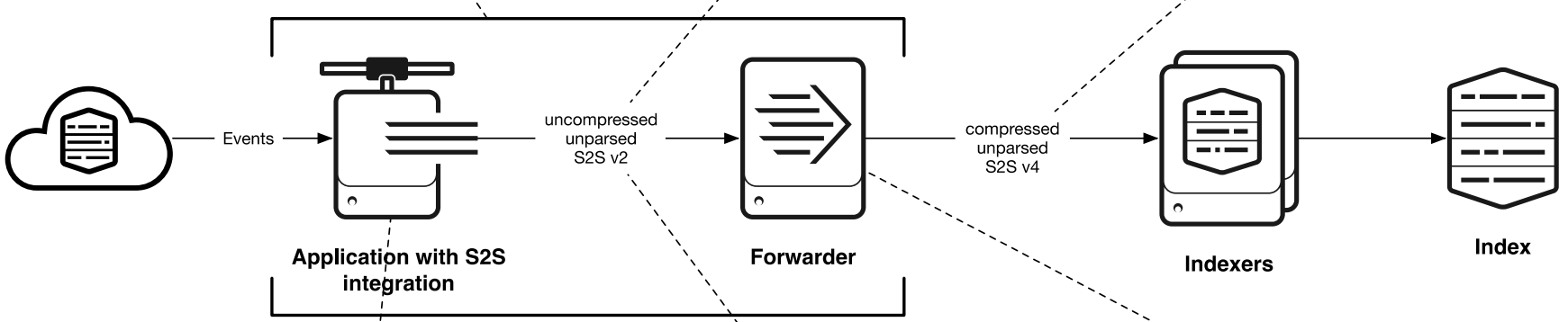
Direct Splunk to Splunk (S2S) integration

S2S chosen for its low protocol overhead and algorithmic complexity

Logs send directly to the forwarder using a native S2S API.

Events with common metadata fields are batched together to reduce transmission overhead

UF enabled compression to reduce NW utilisation at cost of CPU



Indexed fields are created at the point of event creation and submitted via the S2S v2 API

Application disconnect/ reconnect required to keep UF pipelines populated

UF acting like an S2S protocol aware load balancer and updates protocol version in transit

splunk> .conf18

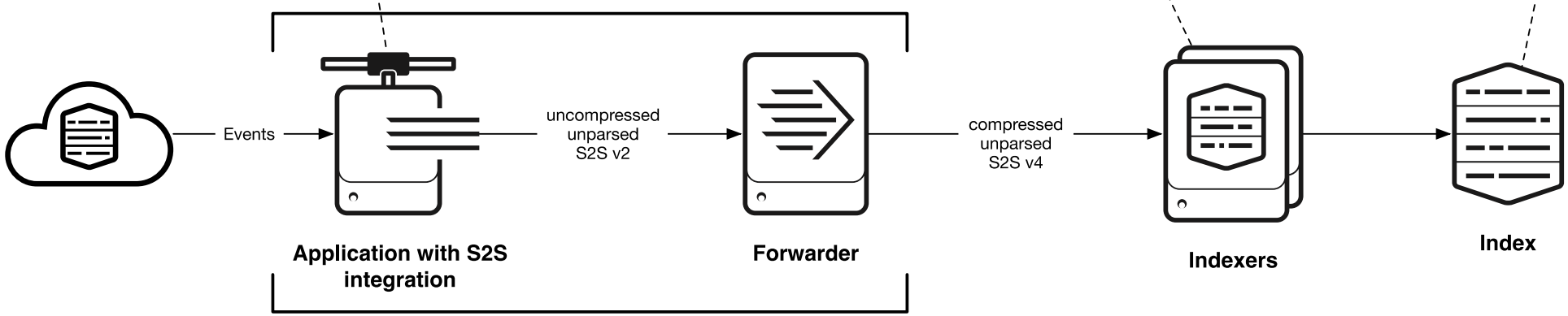
Lowing the cost of indexing

Offload work to event source as much as possible

Meta data extraction, message translation, timestamp parsing, indexed fields all calculated at source

No REGEX
No Punct
No encryption
Time in UTC
No additional indexed fields

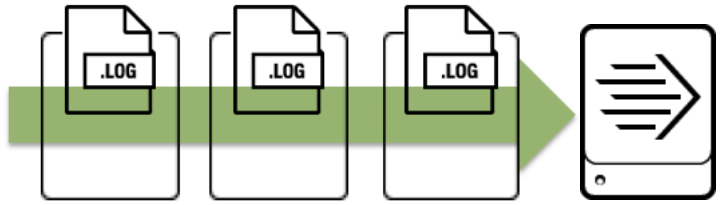
Compression 1:65



Three generations of the logging integration

Application integration v1

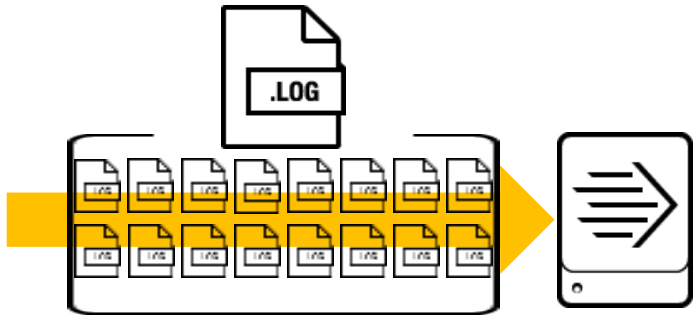
- Each event send individually to the UF
- Metadata applied to each event



Event by event

Application integration v2

- Events bundled together on meta fields
- Reduced CPU load
- Indexing rate doubled
- Optimum S2S payload is 64k (16 for this UC)

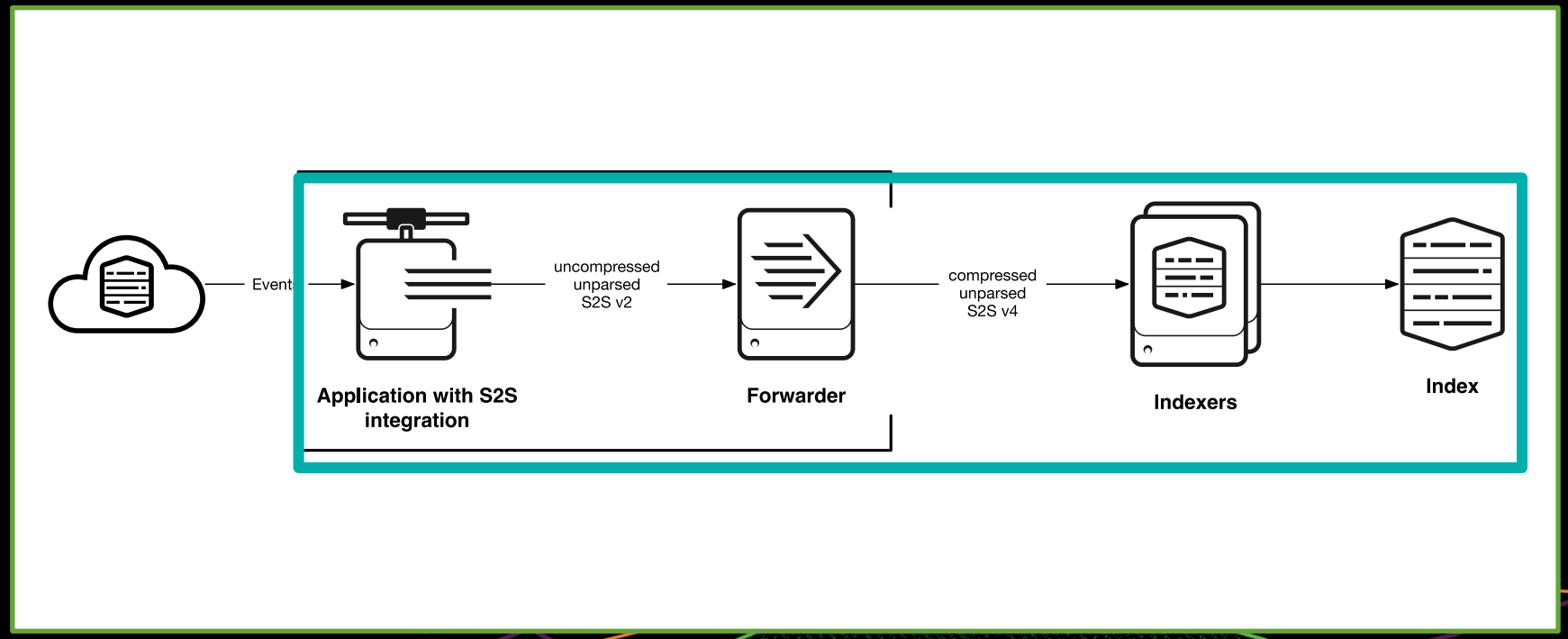


Event bundling

Application integration v3

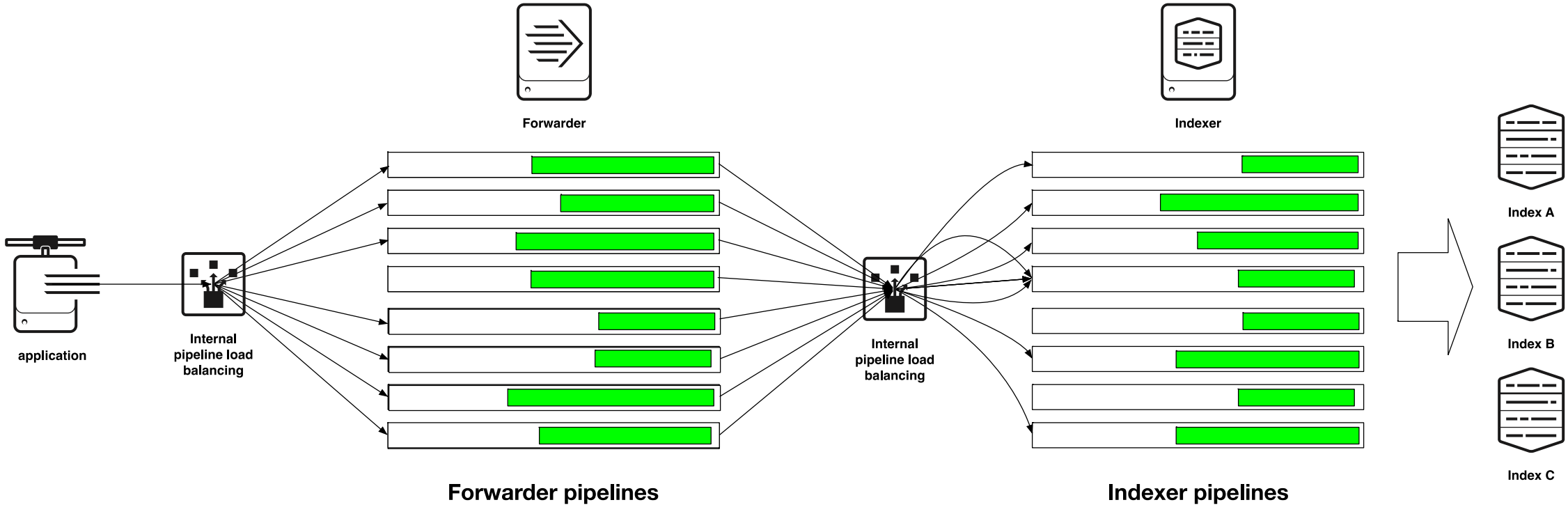
- Application periodic disconnections to improve workload distribution to UF pipelines

“Ingestion Pipelines”



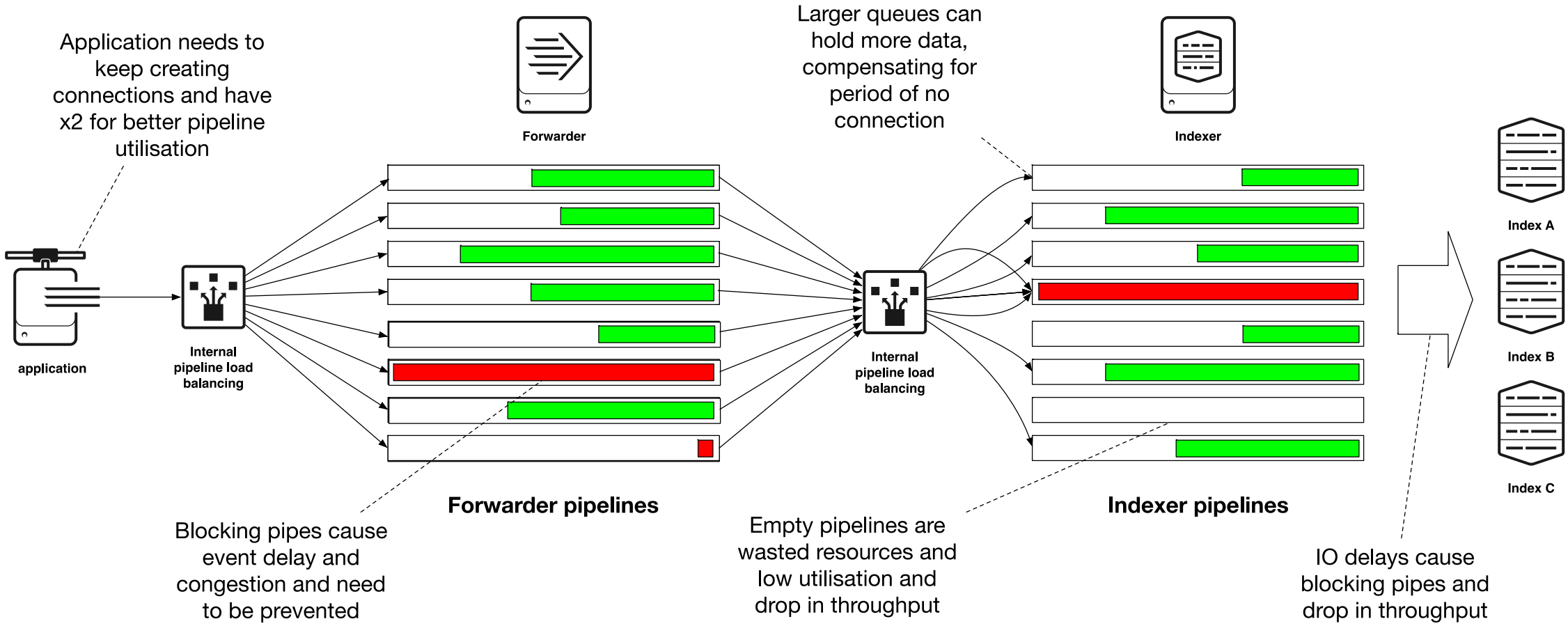
Use multiple pipelines to increase throughput

Each pipeline can process about 20 MB/s with diminishing returns

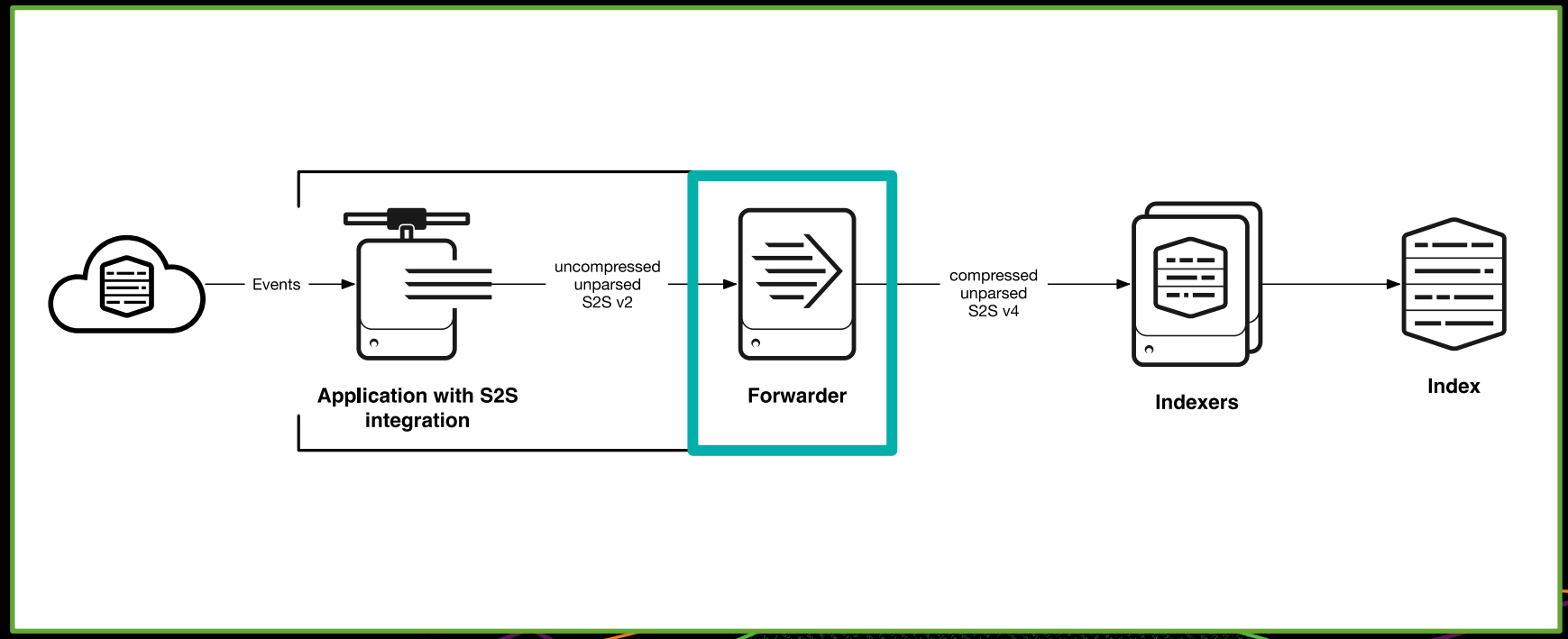


Congestion avoidance via increased randomization

No feedback negotiation, statically randomness only



“Forwarder Tuning”



Multiple Ingestion Pipelines

- Applied the same ingestion pipeline theory to the UFs, turning the UF into multiple UFs without the need for multiple (unsupported) installs
  - Tested between 6 and 12 pipes
- Splunkd changed to run on port 18089
- Forwarders happy 70MB/sec -140MB/sec
  - Ensure to disable thuput throttling

- Limits.conf

```
[thuput]
maxKBps = 0
```

- server.conf

```
[general]
parallelIngestionPipelines = 12
```

- web.conf

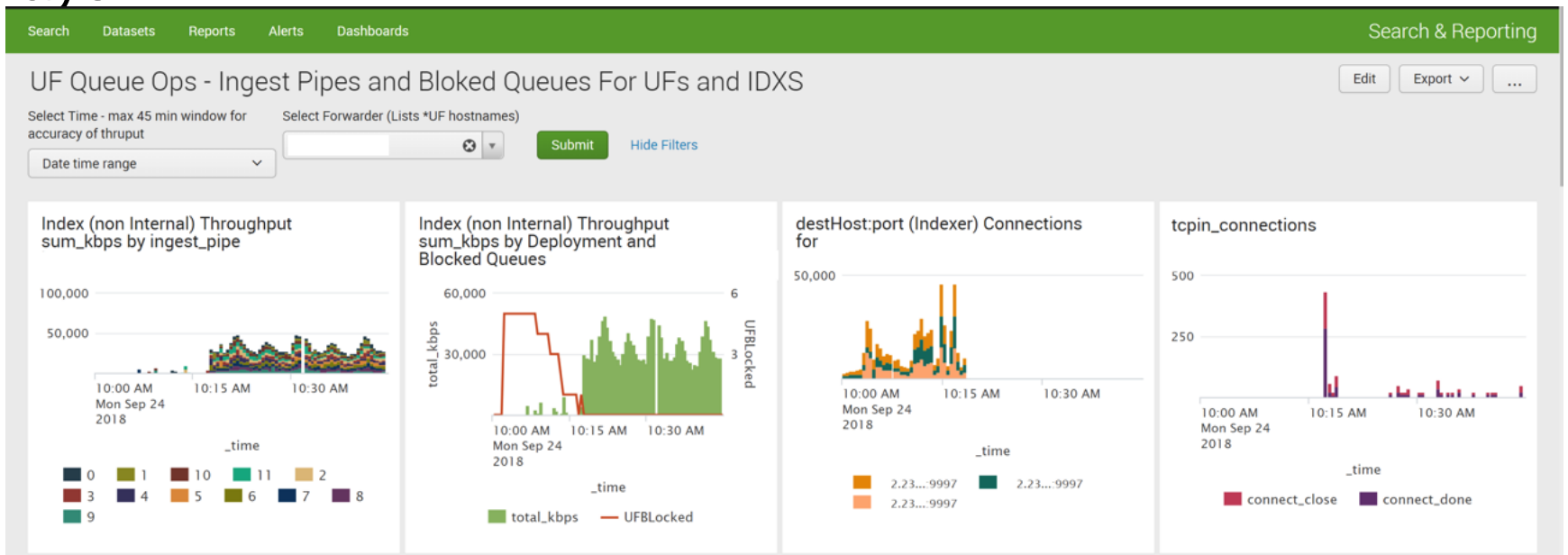
```
# Override 8089 being used by IDX
[settings]
mgmtHostPort = 127.0.0.1:18089
```

- Recommended reading: https://docs.splunk.com/Documentation/Splunk/7.1.2/Capacity/Parallelization

splunk> .conf18

UF Performance Testing

- Routed UF logs to standalone indexer (shown previously)
- Allowed real time searches of UF performance without impacting indexing layer
- Why? We needed to know as soon as the UFs started to block/stop working



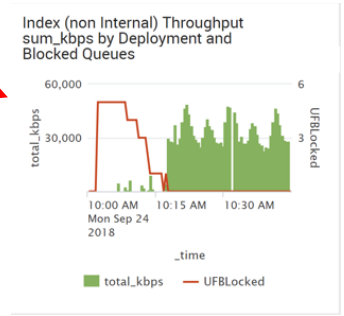
UF Queues Blocking

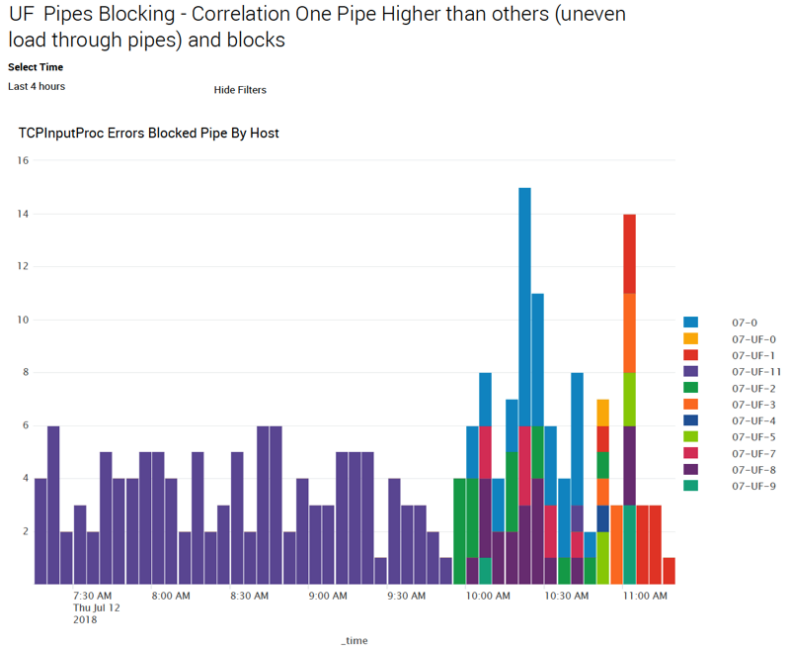
- Smaller dashboard kept on showing UF queues blocking every 4 to 5 minutes
- Search – TCPInput Proc Errors Blocked Pipe By Host

```
index=_internal sourcetype=splunkd host=<UF>
(log_level=ERROR AND ("TcpInputProc - Error
encountered for connection from" AND " Local
side shutting down")) OR (log_level=INFO AND
blocked=true) | eval host-ingest_pipe=host."-
".ingest_pipe | timechart minspan=30sec count
by host-ingest_pipe usenull=false  useother=f
limit=20
```

- Cause
  - Enterprise logging infra connections (12) permanently bound to a pipe
  - Tcpin queue blocked, followed by parsing queue

- Dashboard examples:





splunk> .conf18

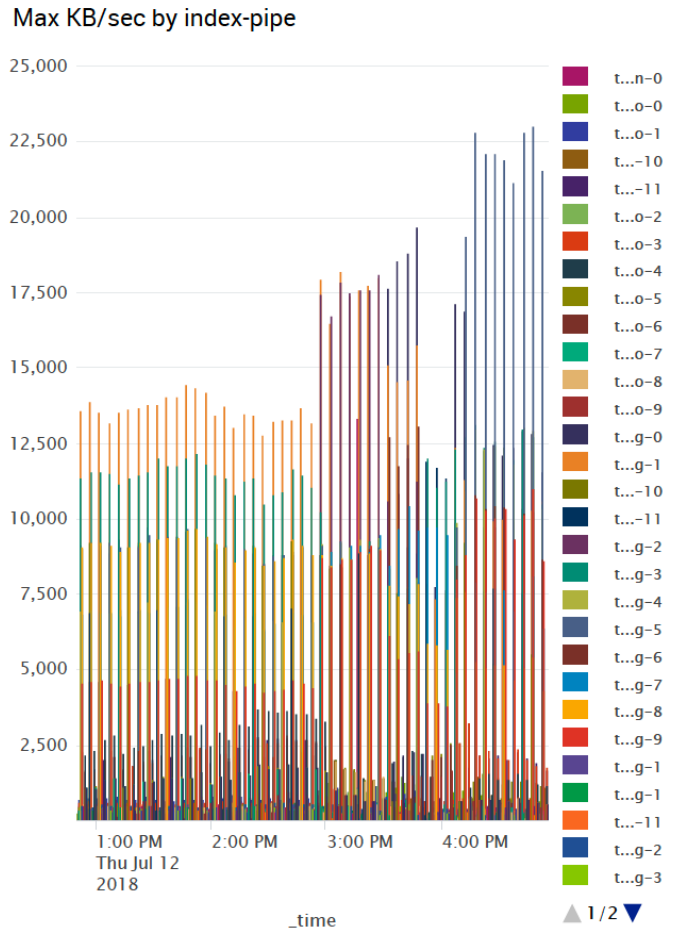
UF Uneven Balance Across Queues – KB/Sec Per Index-Pipe

- Needed more evidence to understand why queues were blocking
- Search – Max KB/sec by index-pipe

```
index=_internal source="*metrics.log*"
host=<UF>  group=per_index_thruput
ingest_pipe=* NOT series=_* | eval index-
pipe=series."-".ingest_pipe |  timechart
minspan=30sec  max(kbps) as "Max KB/sec" by
index-pipe useother=f usenull=f limit=50
```

- Cause
  - 1 connection/pipe sustained > 20MB/sec thuput

- Dashboard example



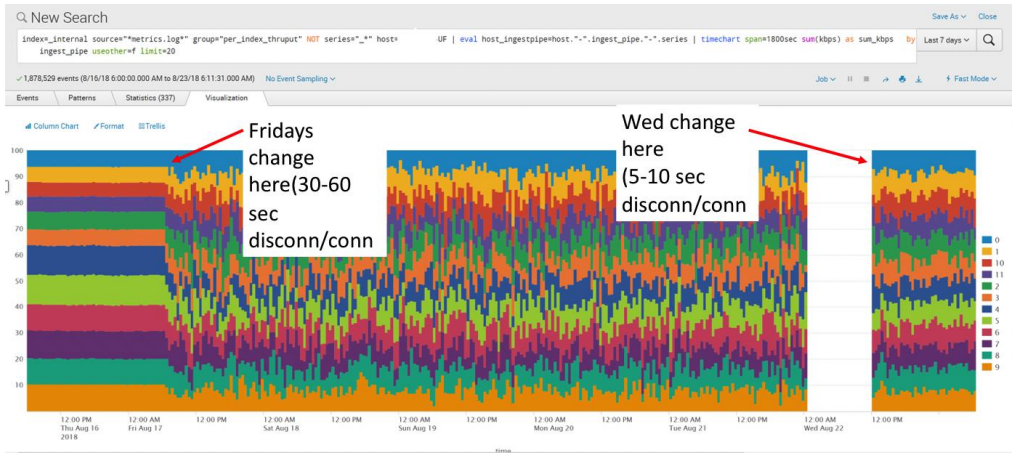
splunk> .conf18

UF Uneven Balance Across Queues Fix

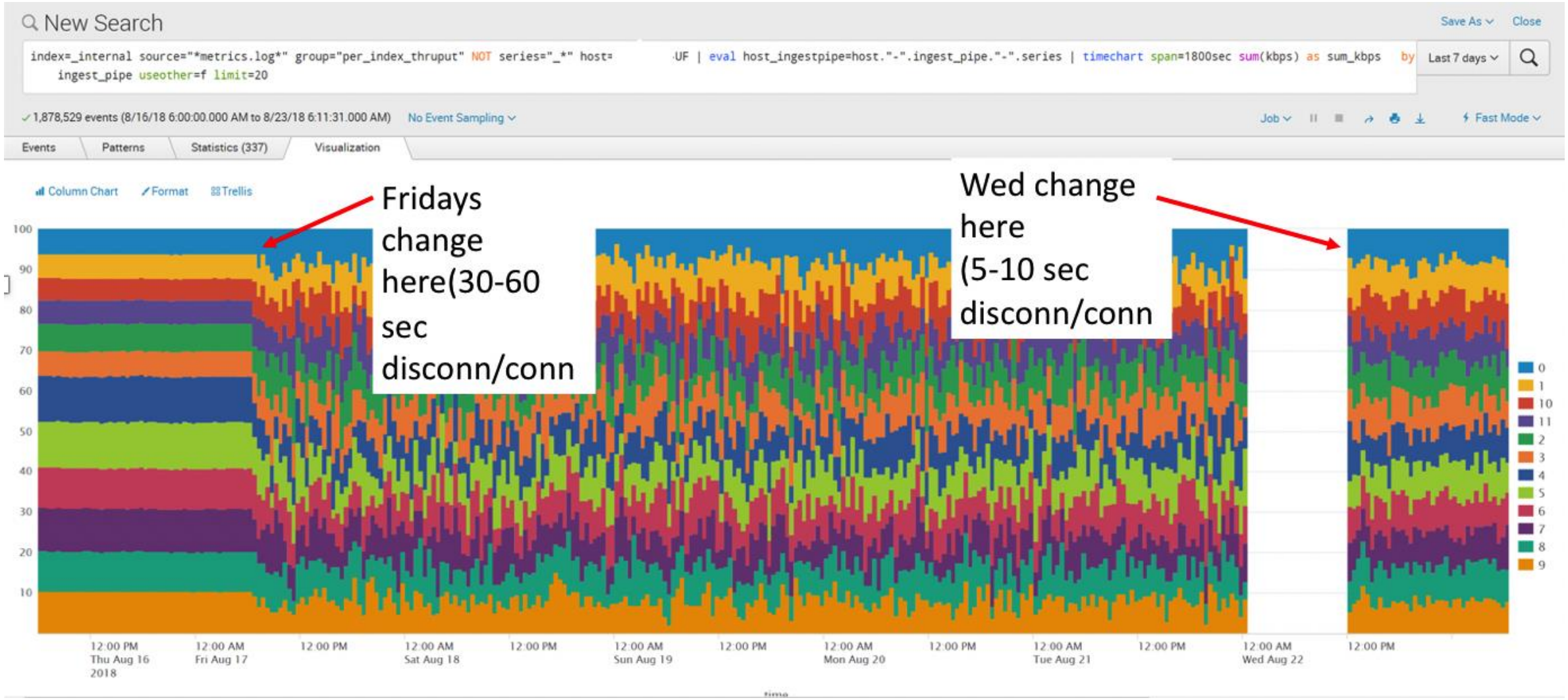
- Enterprise logging infra originally maintained connections to UF on startup
- Implemented enterprise logging infra disconnect from/connect to UF
- Code updated to disconnect/connect on a timer as the dashboard shows

- Index Throughput sum_kbps by ingest_pipe for host

```
index=_internal source="*metrics.log*"
group="per_index_thruput" NOT
series="_*" host=<UF> | eval
host_ingestpipe=host."-".ingest_pipe."-
".series | timechart  sum(kbps) as
sum_kbps   by ingest_pipe useother=f
limit=20
```



UF Uneven Balance Across Queues Fix

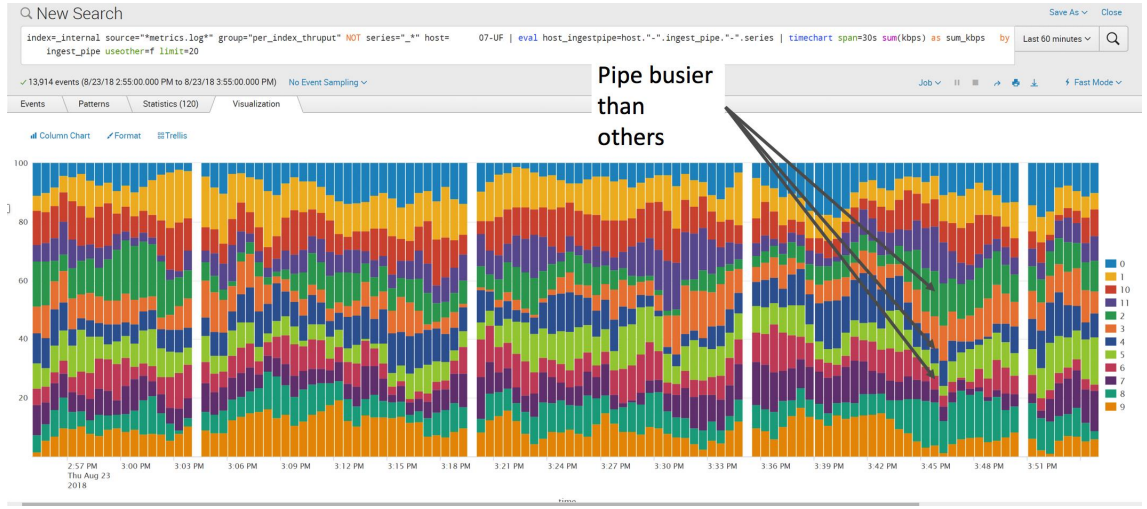


splunk> .conf18

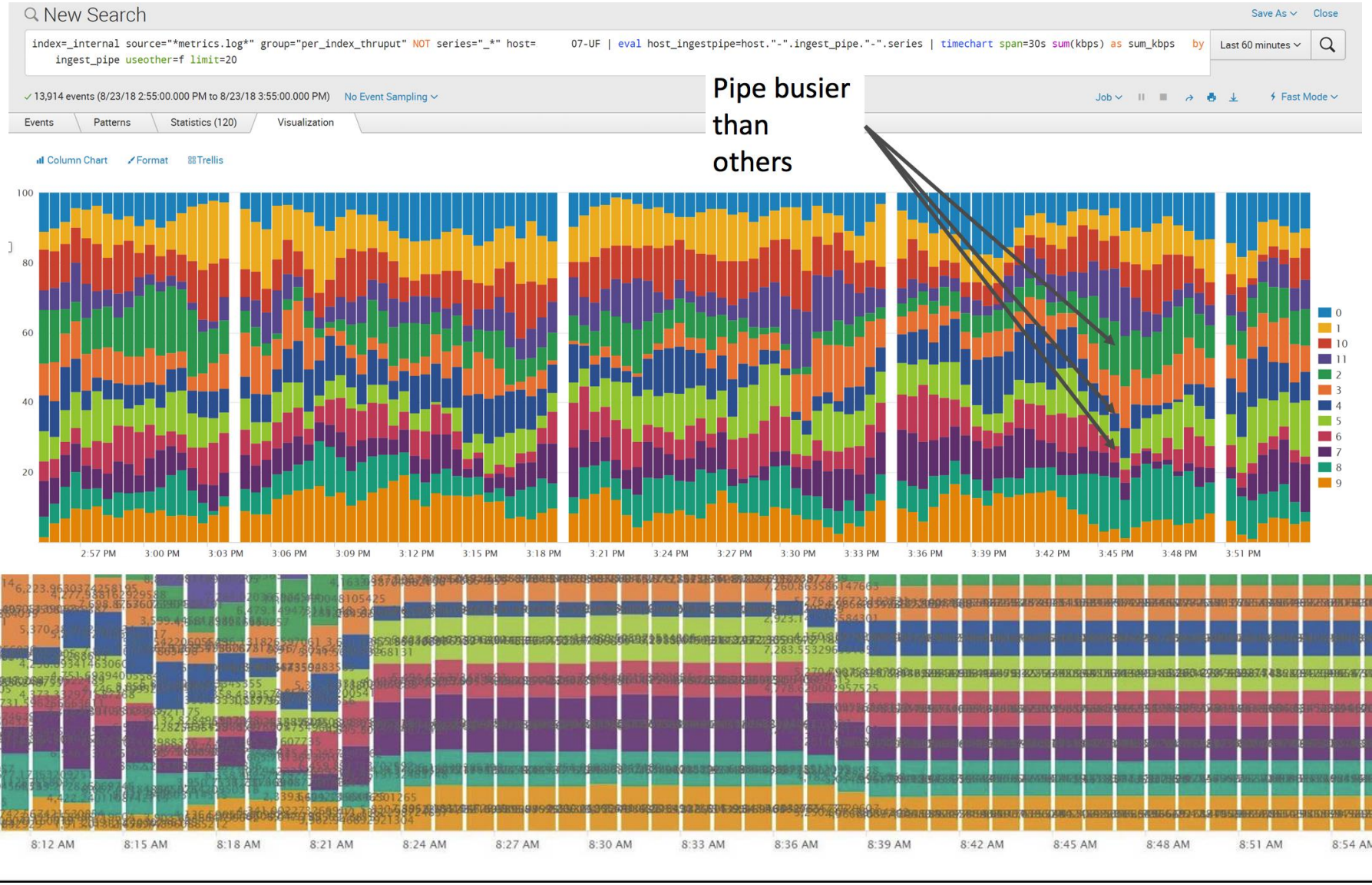
UF Uneven Balance Across Queues - Ingest Pipes

- Disconnect/connect was still causing pipes to be consistently higher than others
- Blocking could still occur..
- ½ the pipes doing the majority of the work
- After further discussions and more development work, updated code was deployed
- The working solution - a 20 min period (max delay for regular full reconnection), the results speak for themselves (on next dashboard!)

- Dashboard example:



UF Uneven Balance Across Queues - Ingest Pipes



splunk> .conf18

UF Queue Size

- Bundling of events gives CPU efficiency/improvement
- Side effect is building makes balancing from Enterprise logging infra to pipe on forwarder uneven
- 16 messages per bundle, up to 4MB each; 16 messages * 4MB message = 64MB
- Only increase if infrequent spikes in queues – not for consistently high queues
- Applied to Input queue and parsing queue

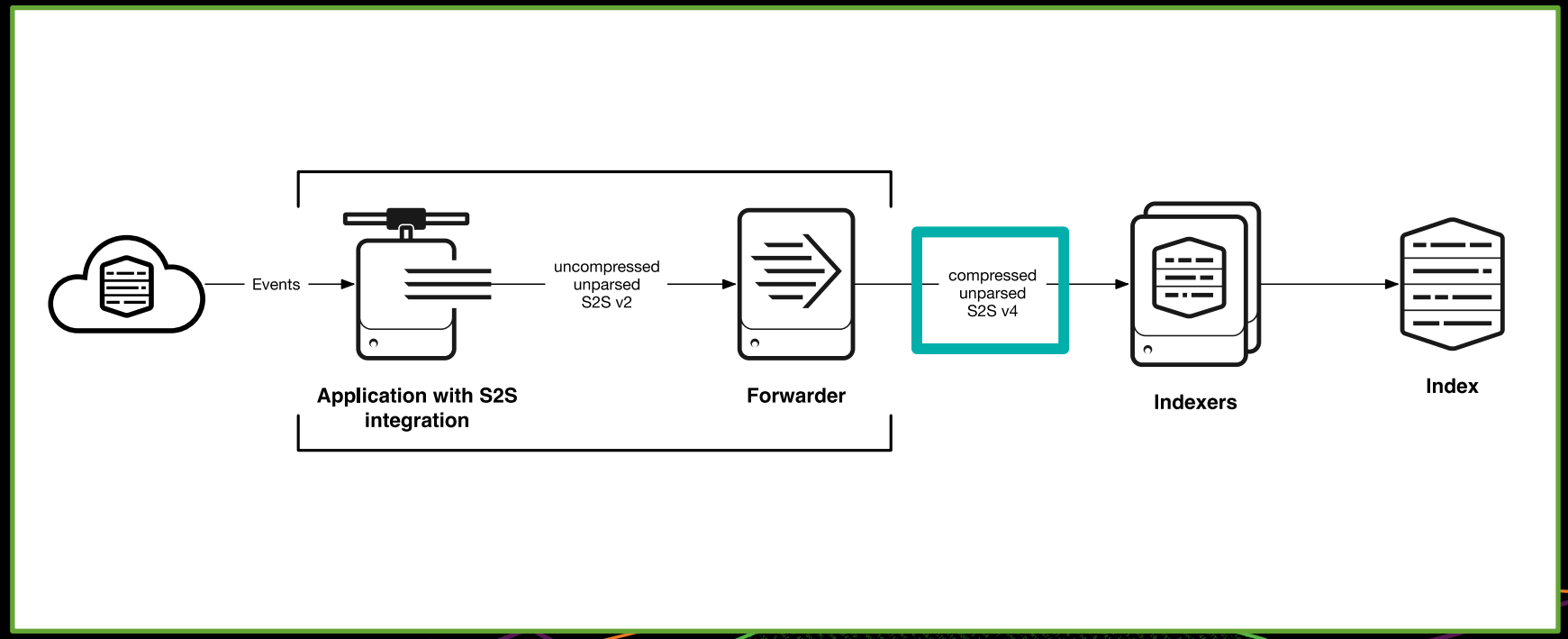
- server.conf:

```
# 16 messages @ 4MB
[queue=parsingQueue]
maxSize = 64MB
```

- inputs.conf:

```
# IDX on same host, use 19077 for
# UF
# 16 messages @ 4MB
[splunktcp://19077]
queueSize = 64MB
```

“Indexer balancing”



Uneven Balancing Across Indexers

- From the monitoring console, Indexing > Performance > Indexing Performance Deployment
- One indexer was doing the work of all the others
- forceTimebasedAutoLB = true possibly the cause?

- Example from MC

Instance	Pipeline Set Count	Indexing Rate (KB/s)	Status	Parsing Queue Fill Ratio (%)	Aggregation Queue Fill Ratio (%)	Typing Queue Fill Ratio (%)	Indexing Queue Fill Ratio (%)
IDX1	8	77739	normal	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.09 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.06 pset3: 0.15 pset4: 0.00 pset5: 0.11 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.06 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00
IDX2	8	47904	normal	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00
IDX3	8	15121	normal	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00
IDX4	8	12	normal	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00
IDX5	8	11	normal	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00	pset0: 0.00 pset1: 0.00 pset2: 0.00 pset3: 0.00 pset4: 0.00 pset5: 0.00 pset6: 0.00 pset7: 0.00

Uneven Balancing Across Indexers

- Fix: Applied on UF
  - outputs.conf:

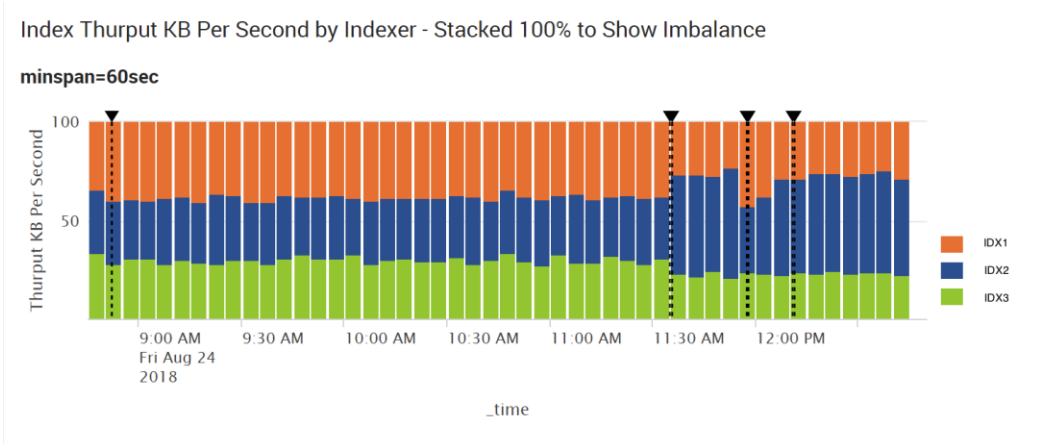
```
[tcpout]
forceTimebasedAutoLB = false
autoLBVolume = 400000000
autoLBFrequency = 15
```

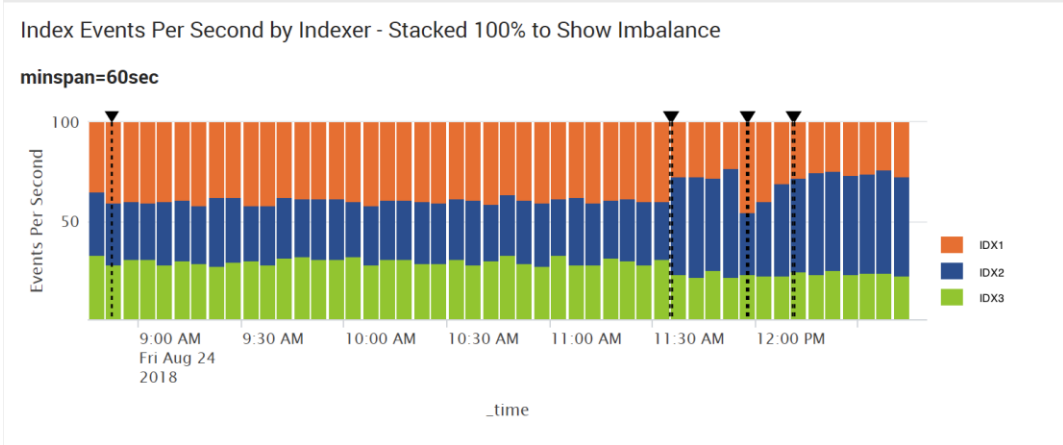
  - props.conf

```
[<sourcetype>]
EVENT_BREAKER_ENABLE = true
EVENT_BREAKER = ([\r\n}+)time=
```

- Load looks better with 5 minute span, but searches are over a ‘earliest=-2m’ to ‘now’

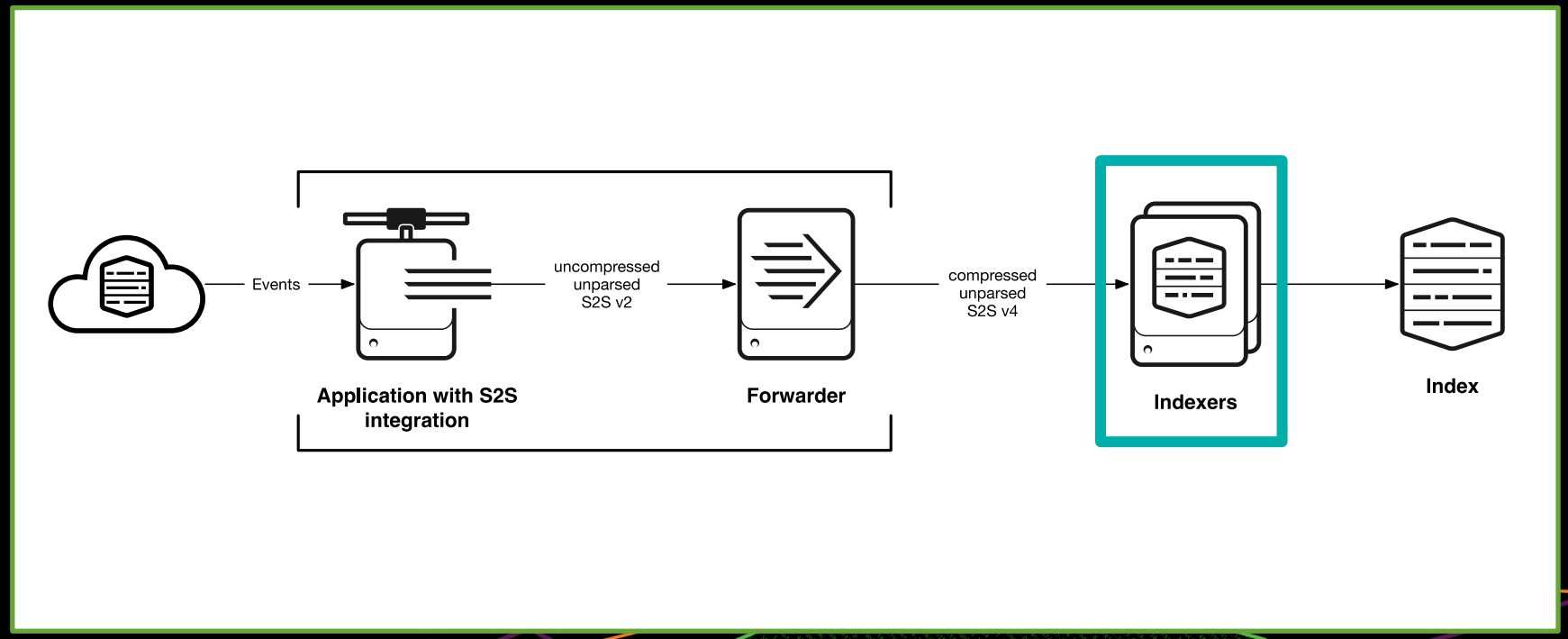
- More from metrics.log





splunk> .conf18

“Indexer Tuning”



Multiple Ingestion Pipelines

- Tuning parallelIngestionPipelines improved use of additional CPU/RAM, turning the IDX into multiple IDXs without the need for multiple (unsupported) installs
  - Tested with 6 and 12 pipes
- Default indexThreads is 8!
- Resulted in doubling the throughput/indexing rate per IDX (including co-hosted UF) achieved during Proof Of Concept
- Benefit: Reduction in server count and costs by ½ (x00 servers * $y0,000)
- 70MB/sec sustained
- Recommended reading: https://docs.splunk.com/Documentation/Splunk/7.1.2/Capacity/Parallelization

- server.conf

```
# Increase for > 20MB/sec, set to
# 12
[general]
parallelIngestionPipelines = 12
```

- indexes.conf

```
# Override default maximum of 8
# pipes on indexers, max is 16
[default]
indexThreads = 16
```

splunk> .conf18

Props Tuning

- All of the tuning and performance improvements enabled us to co-host the Enterprise logging host with splunktcp (S2S) and co-hosted UF/IDX
- Provides best return on hardware (utilizing CPU/RAM)
- Achieved double the throughput/indexing rate per IDX (including co-hosted UF) achieved during Proof Of Concept
- Benefit: Reduction in server count and costs by ½ (x00 servers * $y0,000)

- props.conf

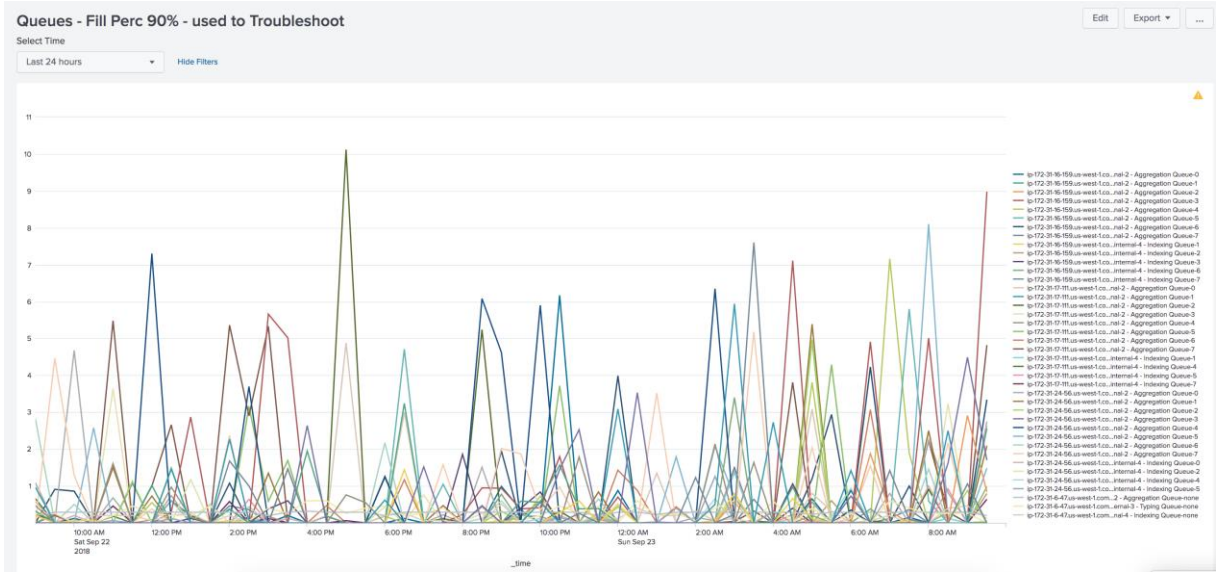
```
[<sourcetype>]
# magic 6 can improve to 4 times
TIME_PREFIX = ^time=
TIME_FORMAT = %s%3N
MAX_TIMESTAMP_LOOKAHEAD = 13
LINE_BREAKER = (\n)time=
SHOULD_LINEMERGE = false
TRUNCATE = 4000000
# not a magic 6 but important
MAX_EVENTS=8192


# More efficiency
KV_MODE = none
ANNOTATE_PUNCT = false
```

splunk> .conf18

Indexer Queues

- As we added more pipes, we needed to keep an eye on the queues - checked frequently
- Running at 50% is ok, shows Splunk is working hard!
- Search (similar to Monitoring Console) – fill perc 90%

```
index=_internal host=* source=*metrics.log
sourcetype=splunkd group=queue (name=tcpin_queue OR
name=parsingqueue OR name=aggqueue OR
name=typingqueue OR name=indexqueue) | eval
name=case(name=="aggqueue","2 - Aggregation
Queue",name=="indexqueue","4 - Indexing
Queue",name=="parsingqueue","1 - Parsing
Queue",name=="typingqueue","3 - Typing Queue") | eval
ingest_pipe = if(isnotnull(ingest_pipe), ingest_pipe,
"none") | search ingest_pipe=* | eval
max=if(isnotnull(max_size_kb),max_size_kb,max_size)
| eval
curr=if(isnotnull(current_size_kb),current_size_kb,cu
rrent_size)  | eval fill_perc=round((curr/max)*100,2)
| eval name=host."-".name."-".ingest_pipe | timechart
Perc90(fill_perc) by name useother=false limit=40
usenull=f
```



“Challenges”

“too many tsidx files” Errors

- “…splunk-optimize process will run intermittently, merging index files together to optimize performance when searching the data…”
- Search:

```
index=internal sourcetype=splunkd “too many tsidx”
```

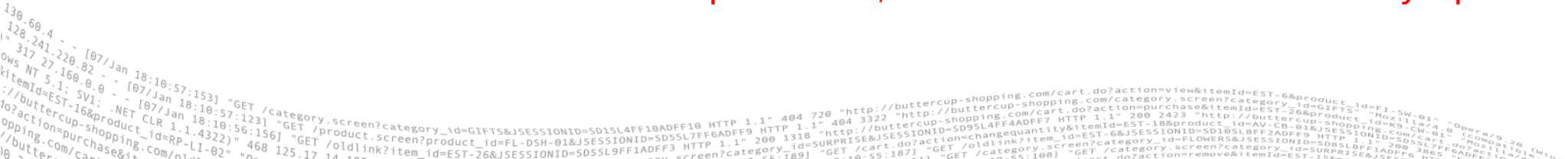
- What we saw in splunkd.log

```
idx=foo Throttling indexer, too many tsidx files in bucket='/opt/splunk_data/hot/foo/db/<hot_bucket>'. Is splunk-optimize working? If not, low disk space may be the cause.
```

- Fix: indexes.conf – for 12 pipelines:

```
# 8 processes per pipeline, 12
# pipelines , set to 12*8=96
[default]
maxRunningProcessGroups = 96

# Set to 6*12 for 12 pipelines
[<per index>]
maxConcurrentOptimizes = 72

# Increase memory usage in line with
# high ingestion rate
[<per index>]
maxMemMB = 20
```

- DISCLAIMER: This is an advanced parameter; do NOT set unless instructed by Splunk Support



splunk> .conf18

"unexpected rc=-9" "from st_txn_put" Errors

- Raised support case for investigation, Support unable to replicate as unique to customer data format
- What we saw in splunkd.log

```
07-12-2018 11:24:17.355 +0000 ERROR STMgr -
dir='/opt/<mount_point>/hot/<index>/hot_v1_4
37' unexpected rc=-9 (kw= <index time
field>::OpsProfilePubGatewayU-TCIL-ISO9735-
TEST1-i13-REQ, len=2048) warm_rc[0,2] from
st_txn_put
```

- Only affects 2 of the _meta/index time fields defined in the S2S configuration on the Enterprise logging host

- Impact: Fields/values not being correctly identified in tsidx files…
- Which field?

```
index=_internal sourcetype=splunkd
log_level=WARN OR log_level=ERROR NOT
"multisite"   "unexpected rc=-9" "from
st_txn_put" | rex field=kw
"(?<affected_key>\w+)::" | timechart
count by affected_key usenull=f
```

- Worked with Enterprise logging host developers
- Identified rogue `\0` control character in the string of the field: for example

```
aarandom1\024string
```

Volume Grow – Leave Headroom

- Don’t set volume to max disk size!
- With high throughput, volume will grow just over its setting, before scheduled roll of buckets from warm to cold
- MB/sec * 30 seconds * # pipes = x GB required above volume size
- Example
  - 70MB/sec throughput/indexing rate
  - 6 pipes
  - 70 * 30 * 6 = 12GB headroom

Event Truncating

- If events are bigger than the default settings, events will truncate
- Search: Breaking Event Because of Limit

```
index=_internal sourcetype=splunkd "WARN AggregatorMiningProcessor - Breaking event because limit" | stats last(_time) as LastTime count by message | convert ctime(LastTime) as LastTime
```

- Cause:

```
09-25-2018 13:57:55.097 +0000 WARN AggregatorMiningProcessor - Breaking event because limit of 256 has been exceeded - data_source="<source>", data_host="<host>", data_sourcetype="<sourcetype>"
```

- Default setting in props.conf for the "…maximum number of input lines to add to any event..." is 256, and "…maximum line length…" is 10000bytes
- Some sourcetypes can be 8000+ input lines, and long events
- Fix in props.conf:

```
[<sourcetype>]
MAX_EVENTS = 8192
TRUNCATE = 1500000
```

splunk> .conf18

“Next Steps”

New compression algorithm in 7.2

- Zstandard (or Zstd)
  - is a lossless data compression algorithm
  - developed by Yann Collet at Facebook
- Improved splunkd performance
  - Improved decompression
  - Improved compression for journal
- https://en.wikipedia.org/wiki/Zstandard

indexes.conf

```
[index_name]
journalCompression = gzip|lz4|zstd
```

Workload management

- Workload management lets you:
  - Reserve system resources for search and indexing processes.
  - Prioritize critical search workloads.
  - Prevent over-usage of system resources.
  - Avoid data-ingestion latency due to heavy search load.
  - Create rules to control access to resources based on a

http://docs.splunk.com/Documentation/Splunk/7.2.0/Workl nagement

Enable Parallel Reduce

- The parallel reduce process inserts an intermediate reduce phase into the map-reduce paradigm, making it a three-phase map-reduce operation. In this intermediate reduce phase, a subset of your indexers acts as **intermediate reducers**. The intermediate reducers perform reduce operations on mapped results and perform reduce operations on those results. Reported search commands. When the intermediate reducers finish their work, they send the results to the search head, where the final aggregation operations take place. The parallel reduction work that otherwise would be done entirely by the search head. Faster completion times for high-cardinality searches with large numbers of search results.

Work with larger data sets !!!!

http://docs.splunk.com/Documentation/Splunk/7.1.3/Di allelreduceoverview

Introduce INGEST_EVALS

- 7.2 introduces full EVALS during the ingestion process

- Detect threshold breaching during ingestion
- Implement sampling operations during ingestion
- Enrich data at index time via lookups
- Clone events based on rules

Thank You

Don't forget to rate this session in the .conf18 mobile app