Inputs: File, Network, Script, and More! Splunkd: Pipelines & Processors & Queues, Oh my!

Amrit Bath
Jag Kerai

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.

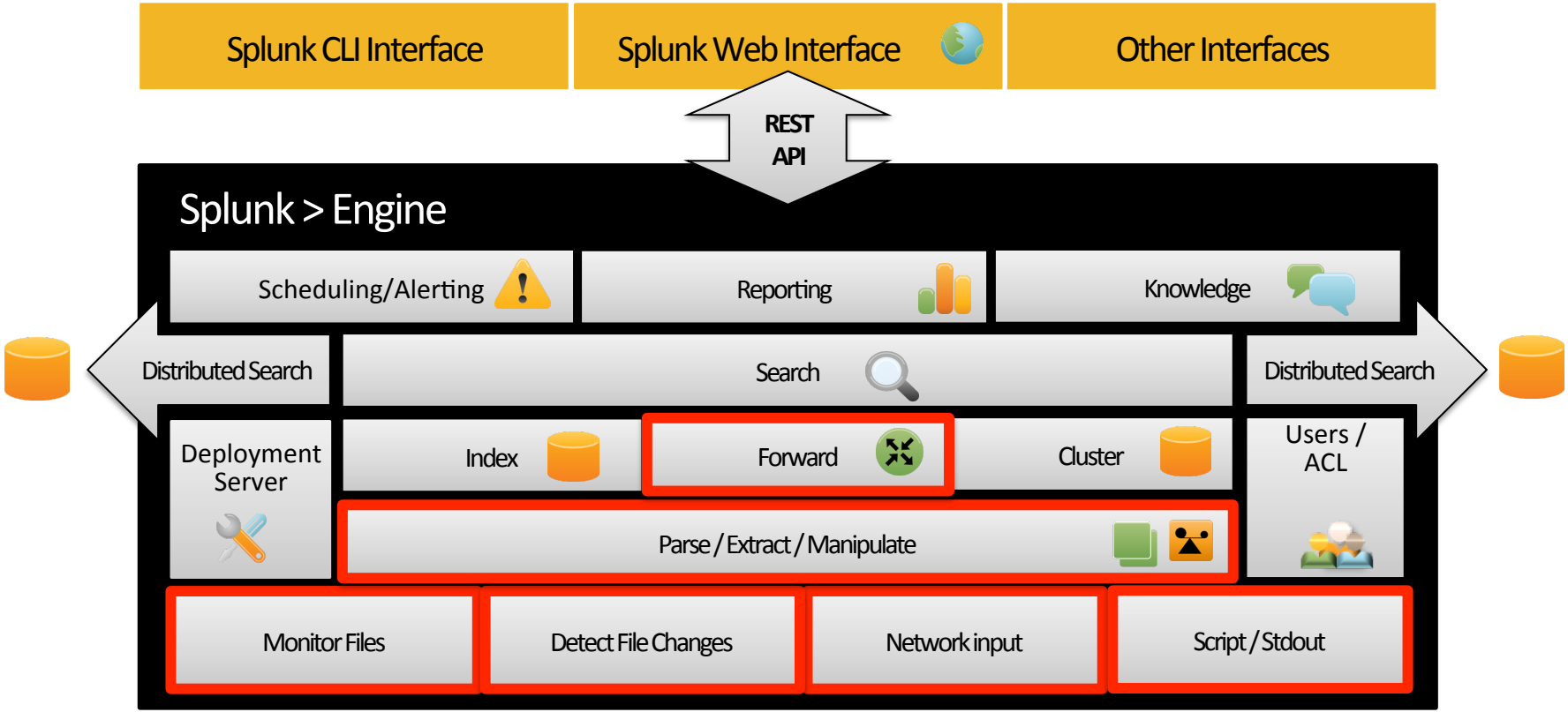
Agenda

- Splunk**d** building blocks:
  - Pipelines, processors, queues
  - Not here: Indexing/clustering, searching.
- Where data goes & how
- Where data comes from
- Debugging/optimizing
  (What Would Octavio Do?)

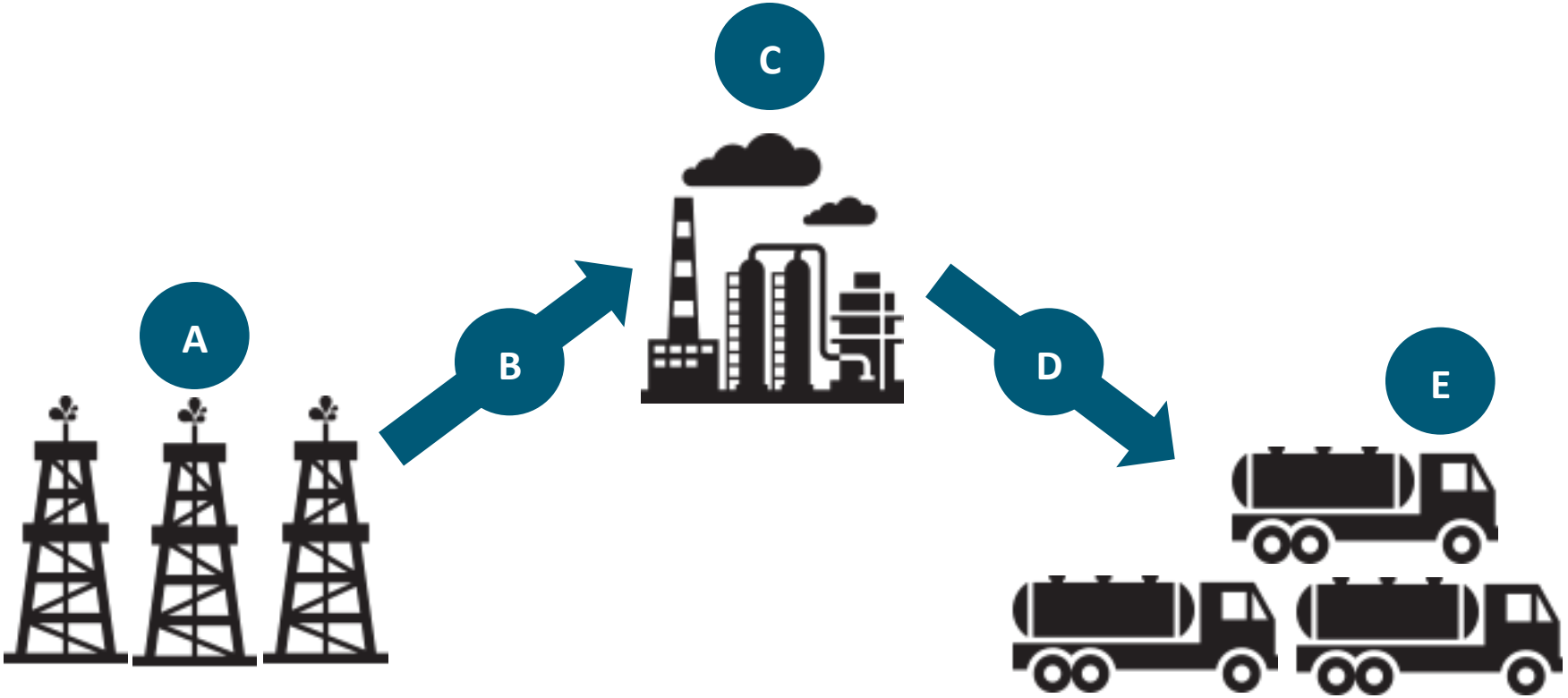
About Us

- Jag Kerai (2008) – Eng & Manager
  - Forwarding/receiving, Splunkd framework, HA/ Clustering, Deployment server, SAML, …
  - Previously: Xsigo Systems, Brocade Communications, SAP Lab – Web apps in 1995!
  - Is a smart dude
- Amrit Bath (2005) – Eng & Hungover
  - CLI, Deploy Serv, Tailing, REST API, Universal Forwarder, Indexed Extractions, Cloud…
  - Previously: Unemployment, College
  - Tries to enjoy working on cars

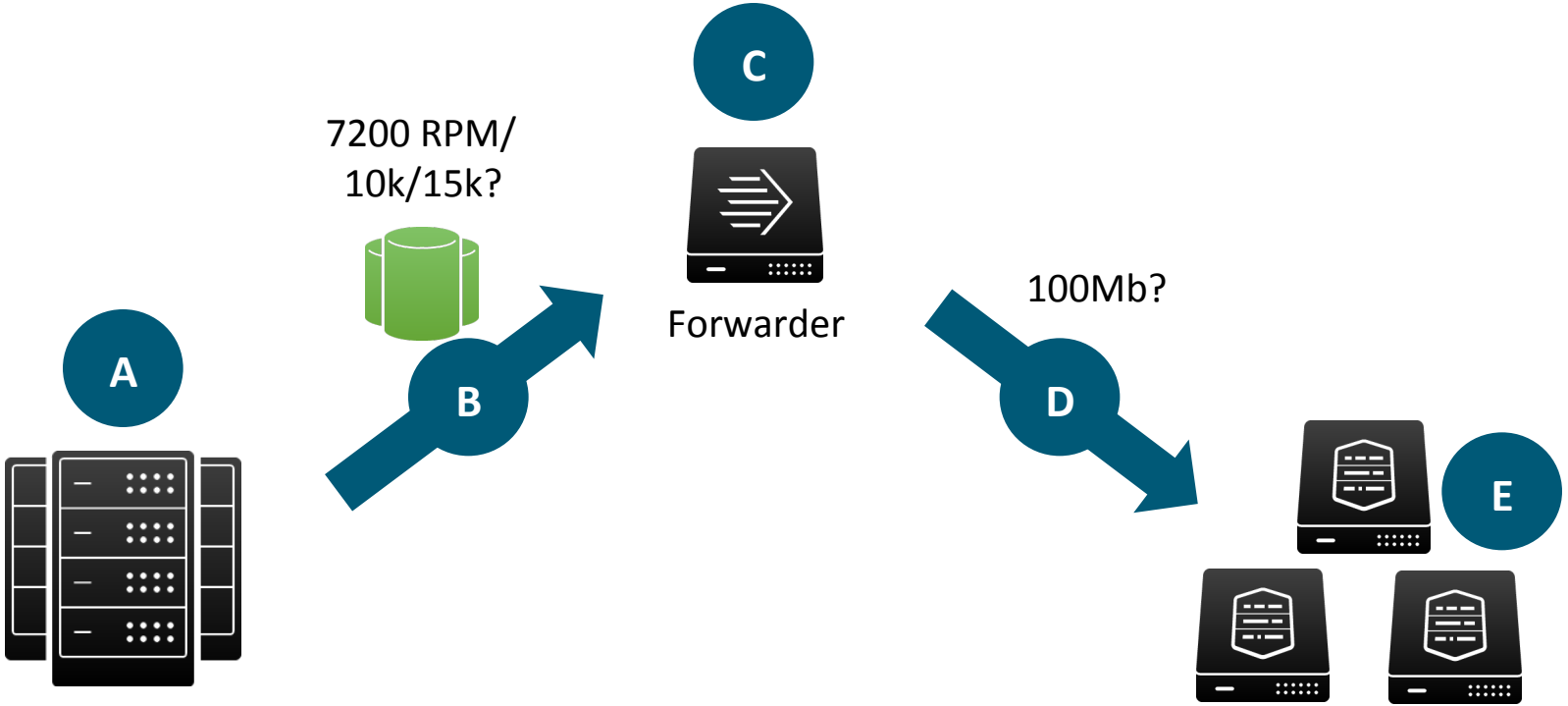
Splunk Architecture



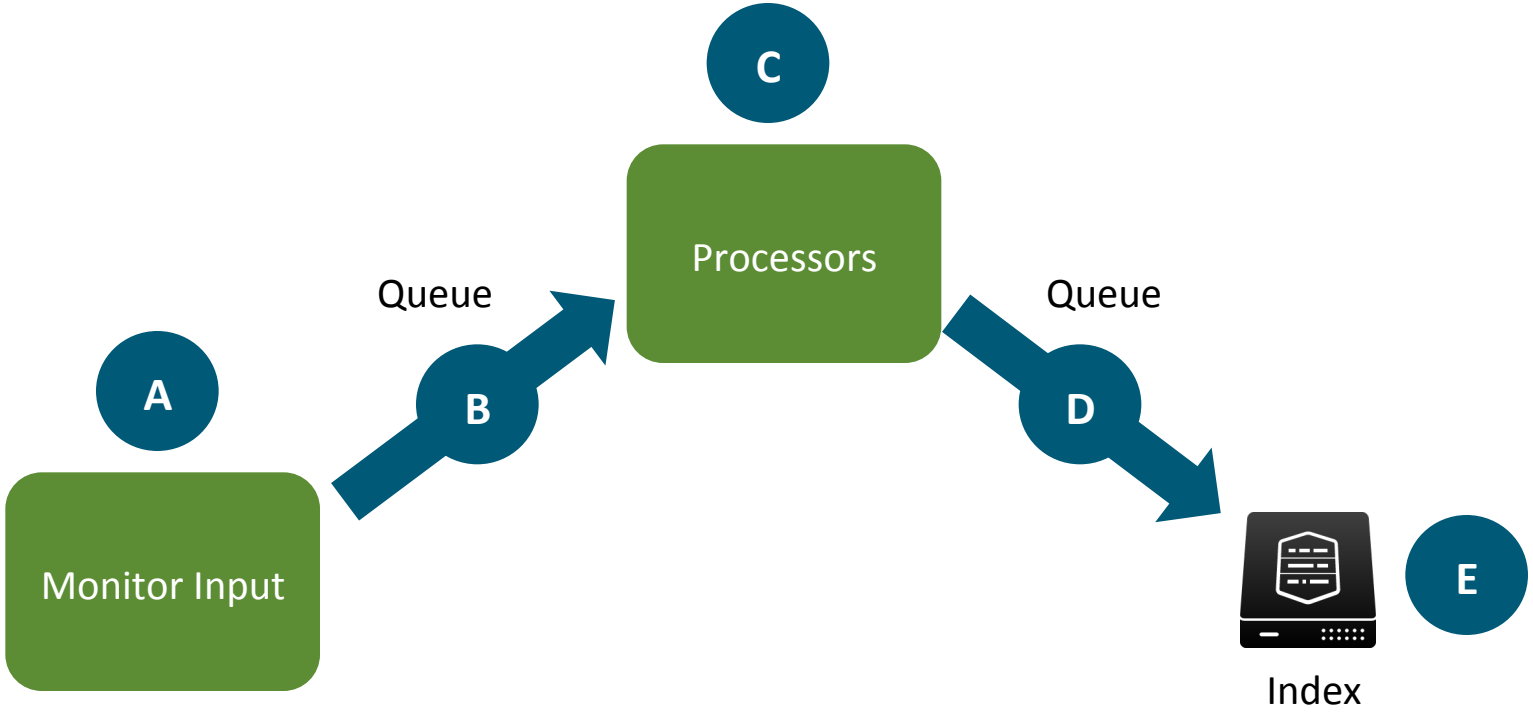
It’s All a Pipeline



Really

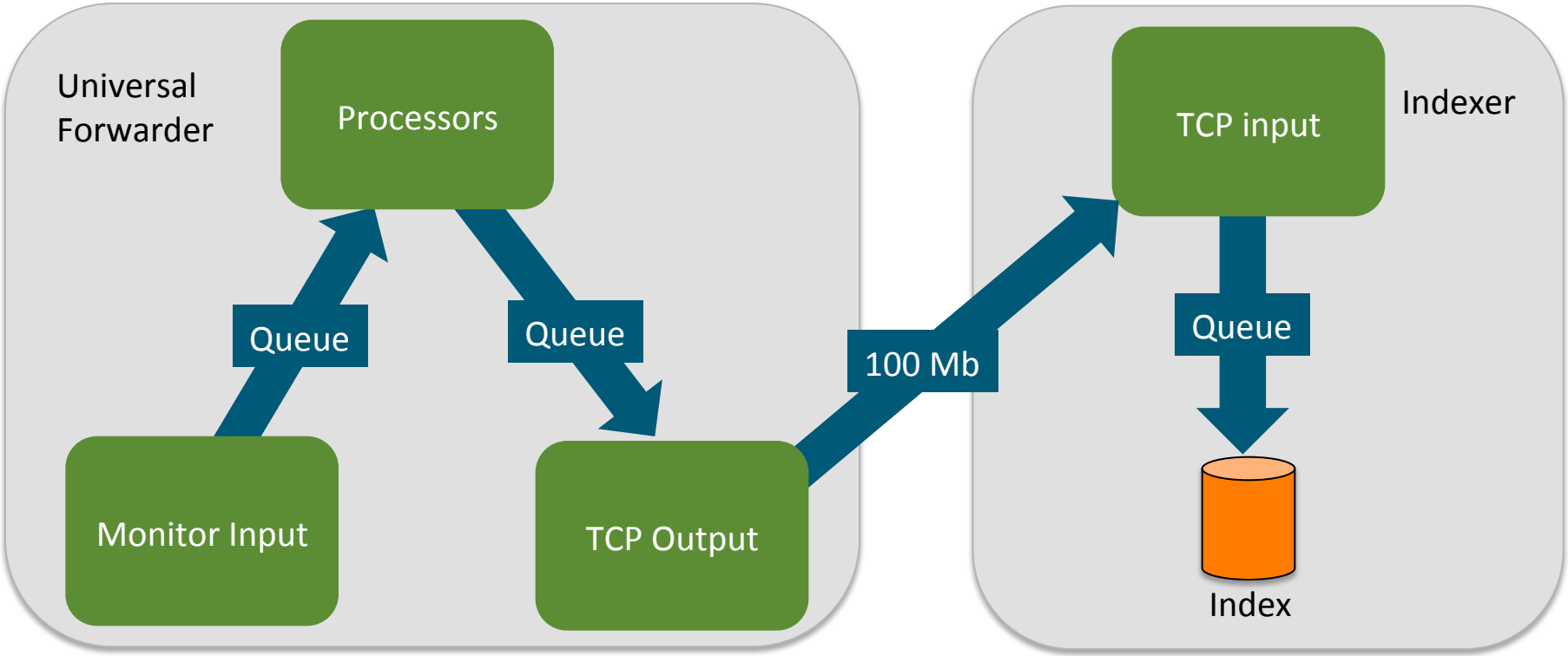


Internally, too!



And Across Multiple Instances!

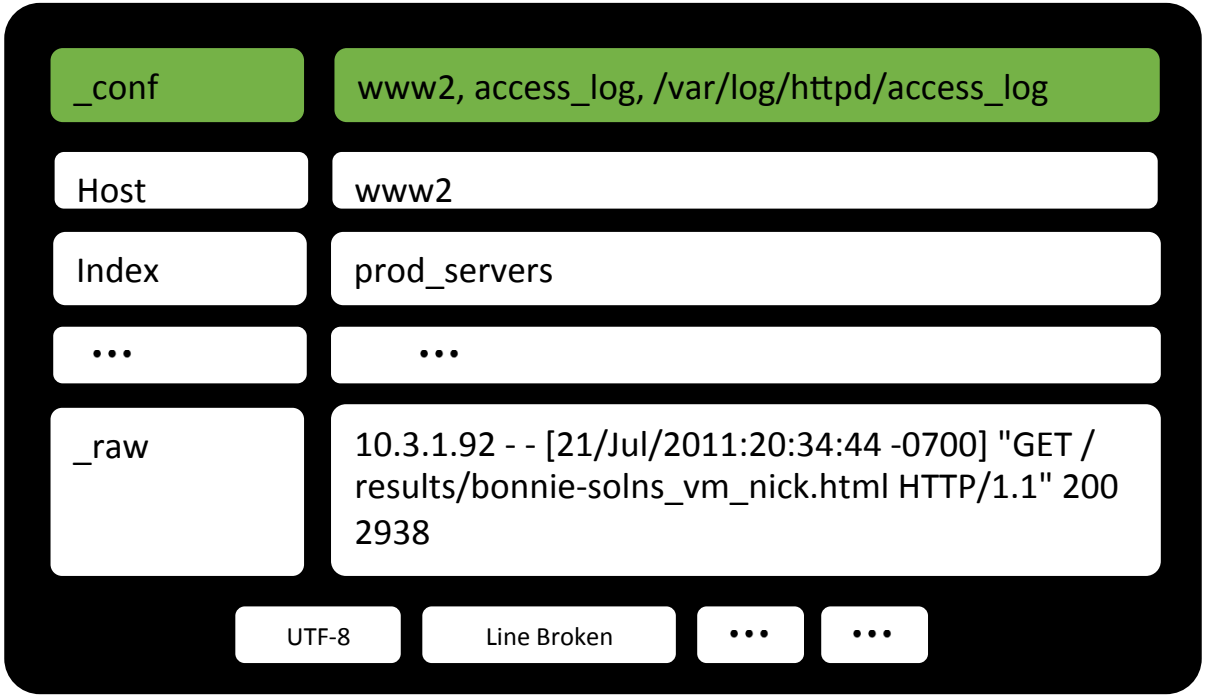
(This is the biggun')



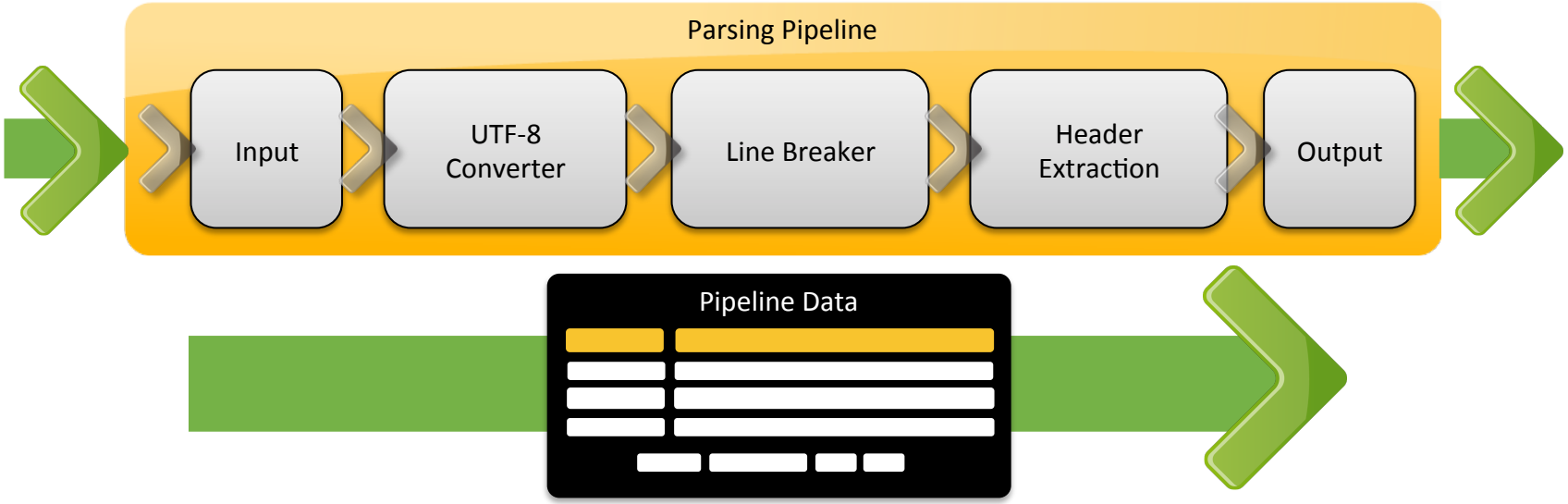
Data Structures & Routing

splunk>

Pipeline Data



Pipelines

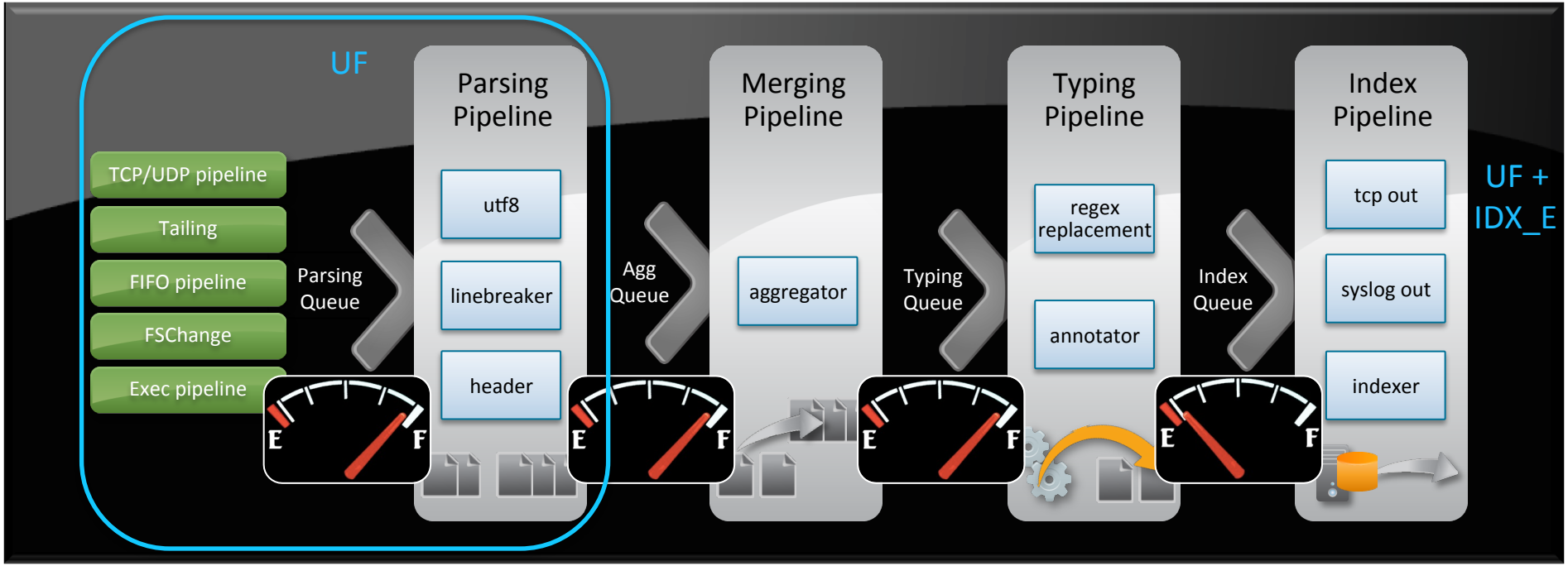


- Pipeline: thread
- Data flows through linearly, hits multiple pipelines before indexing
- Naturally allow parallelism, modularity
- Config files: $SPLUNK_HOME/etc/modules/ Merged: var/run/splunk/composite.xml

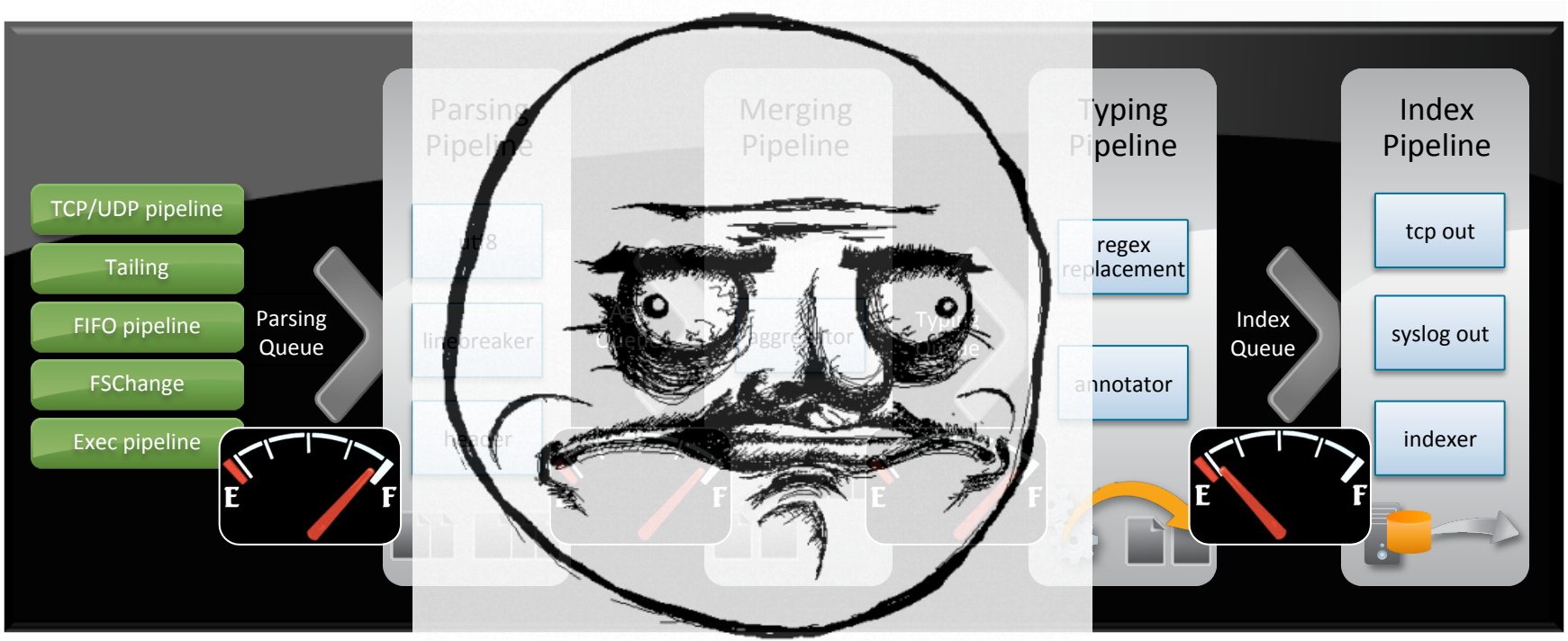
Processor

- Processor: performs small but logical unit of work
- Contained within a pipeline
- Executed by pipeline thread
- Example: LineBreaker, Aggregator, TcpInput, Index

Pipelines/Processors (Debugging)

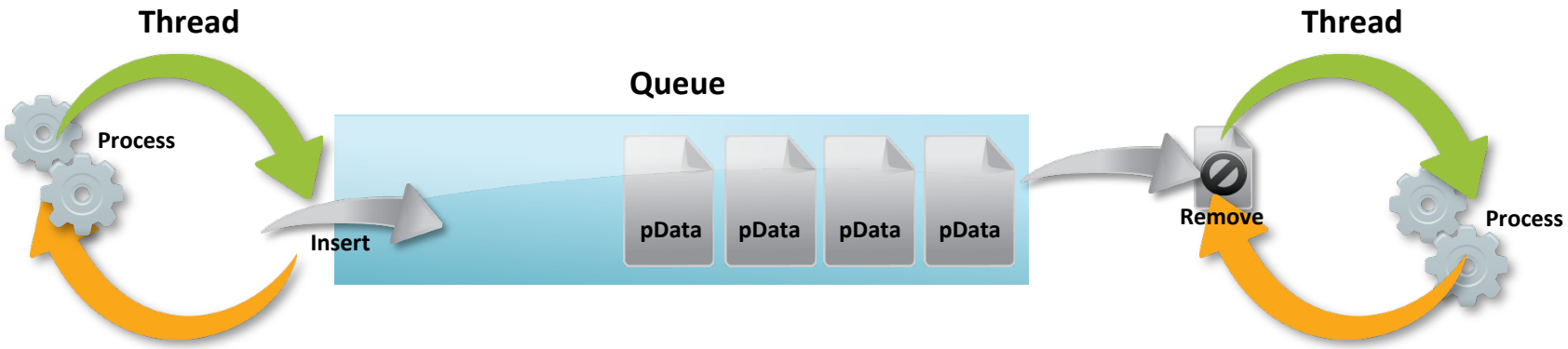


Pipelines/Processors (Debugging)



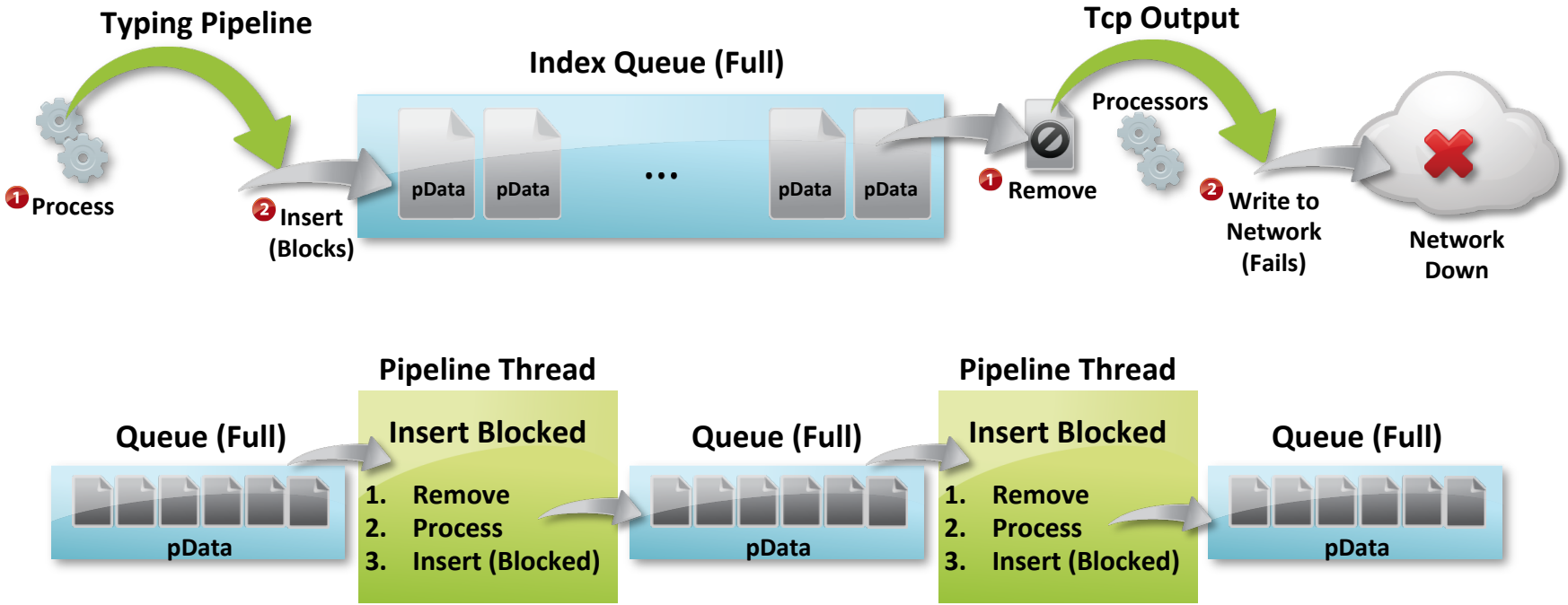
Pipelines/Processors (Debugging)

Queue



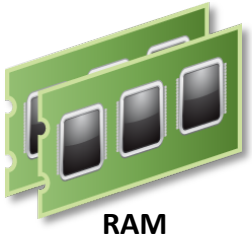
- Queue size bounded by memory
- Variable size pipeline data

Queue


Typing Pipeline
Index Queue (Full)
Tcp Output
Processors
1 Process
2 Insert (Blocks)
pData
pData
pData
pData
1 Remove
2 Write to Network (Fails)
Network Down
Pipeline Thread
Pipeline Thread
Queue (Full)
Insert Blocked
Queue (Full)
Insert Blocked
Queue (Full)
1. Remove
2. Process
3. Insert (Blocked)
1. Remove
2. Process
3. Insert (Blocked)
pData
pData
pData

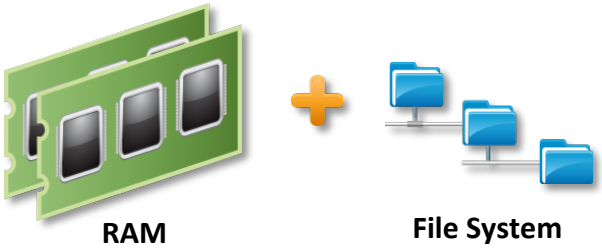
Persistent Queue

Regular Queues



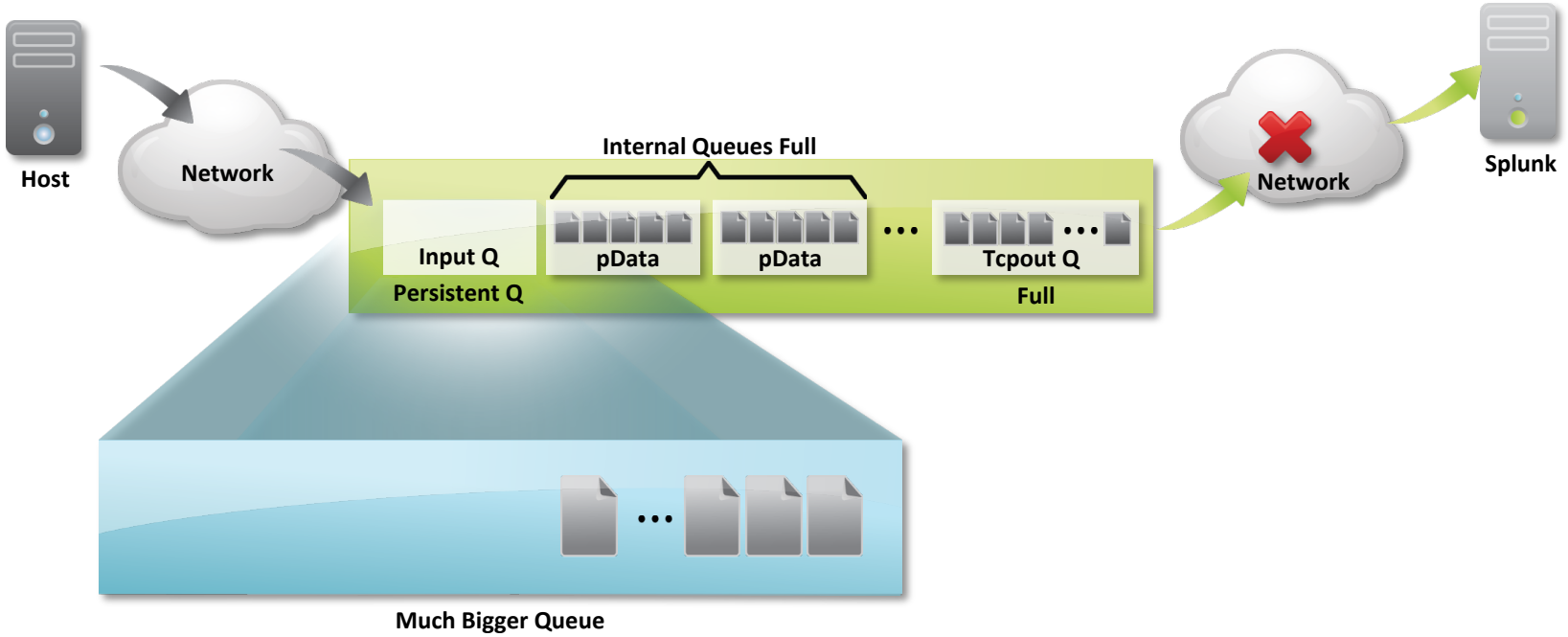
- Writer blocks if Q is full

Persistent Queues



- If memory used up, use file system
- Writer does not block if memory is used up
- Think virtual memory

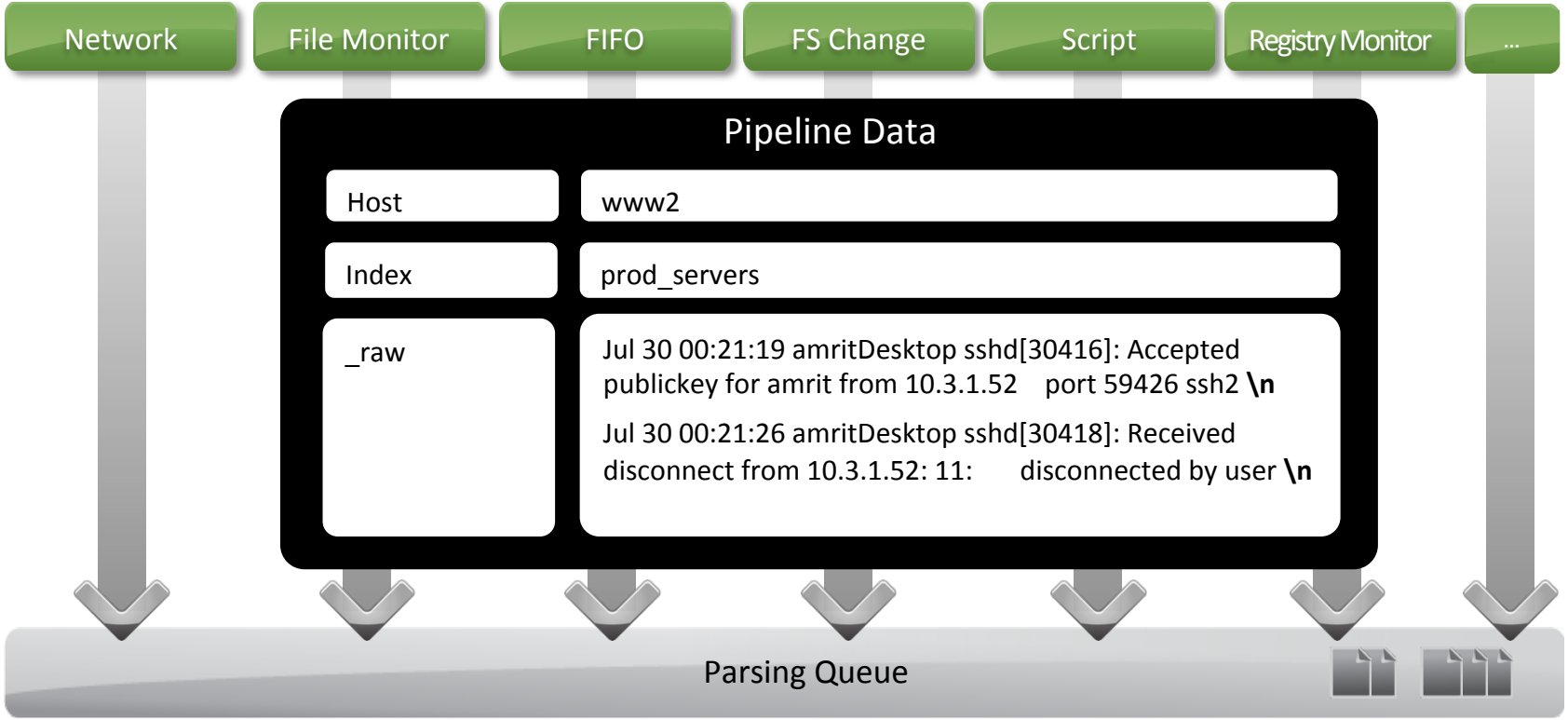
Persistent Queue



Processors: Input & Parsing

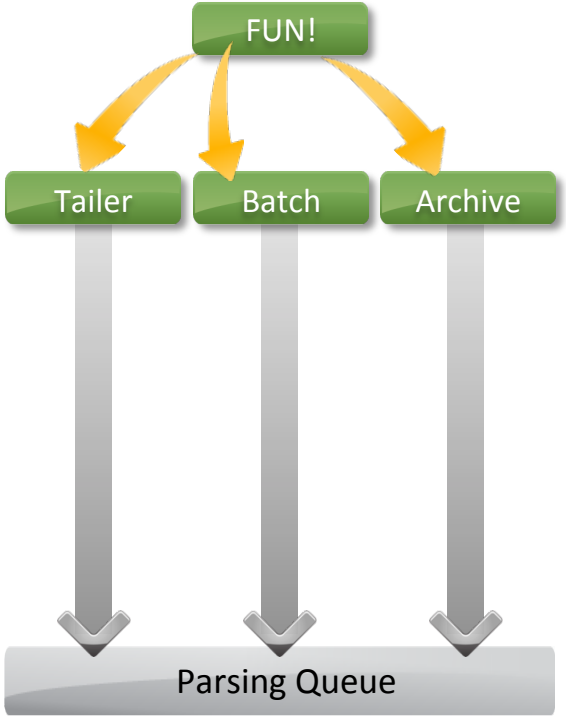
splunk>

Input Pipelines

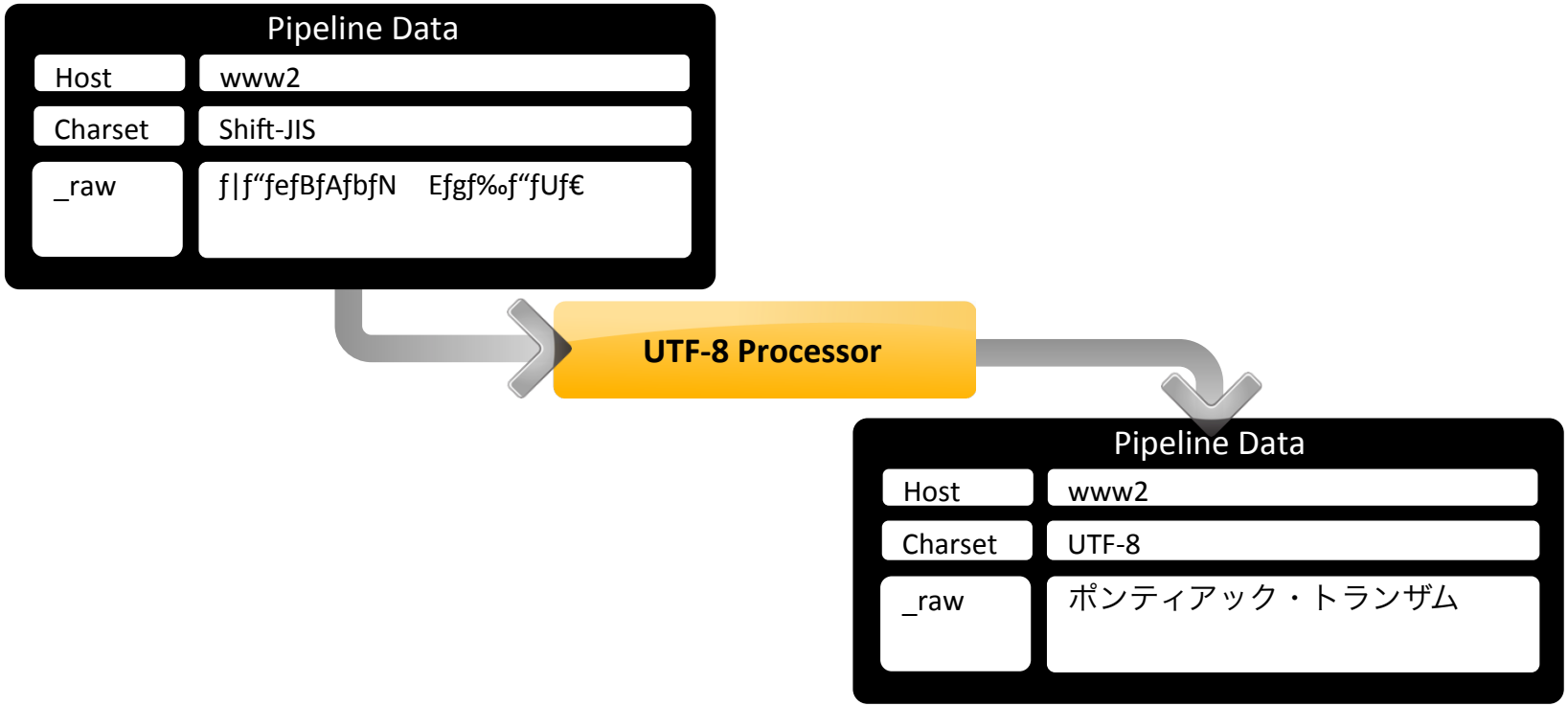


Monitor Input (aka Tailing Processor)

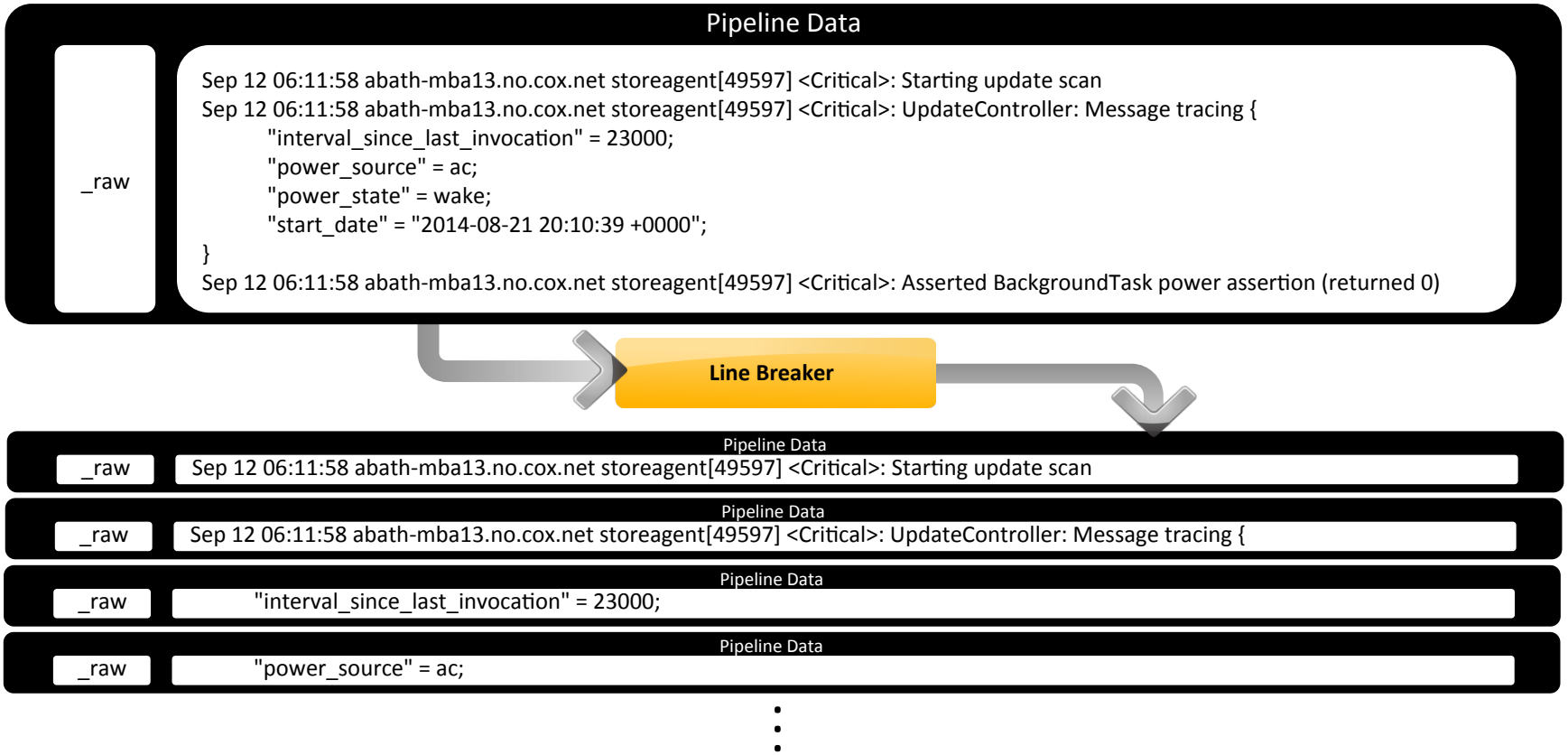
- Two synchronous components:
  - File Update Notification (FUN!)
  - Tailer: reads files
- Files are read:
  1) One at a time
  2) In 64KB chunks,
  3) Until EOF.
  - Can read large files & archives in parallel.
- Send <=64 KB chunks to output queue



Parsing: UTF-8 Processor



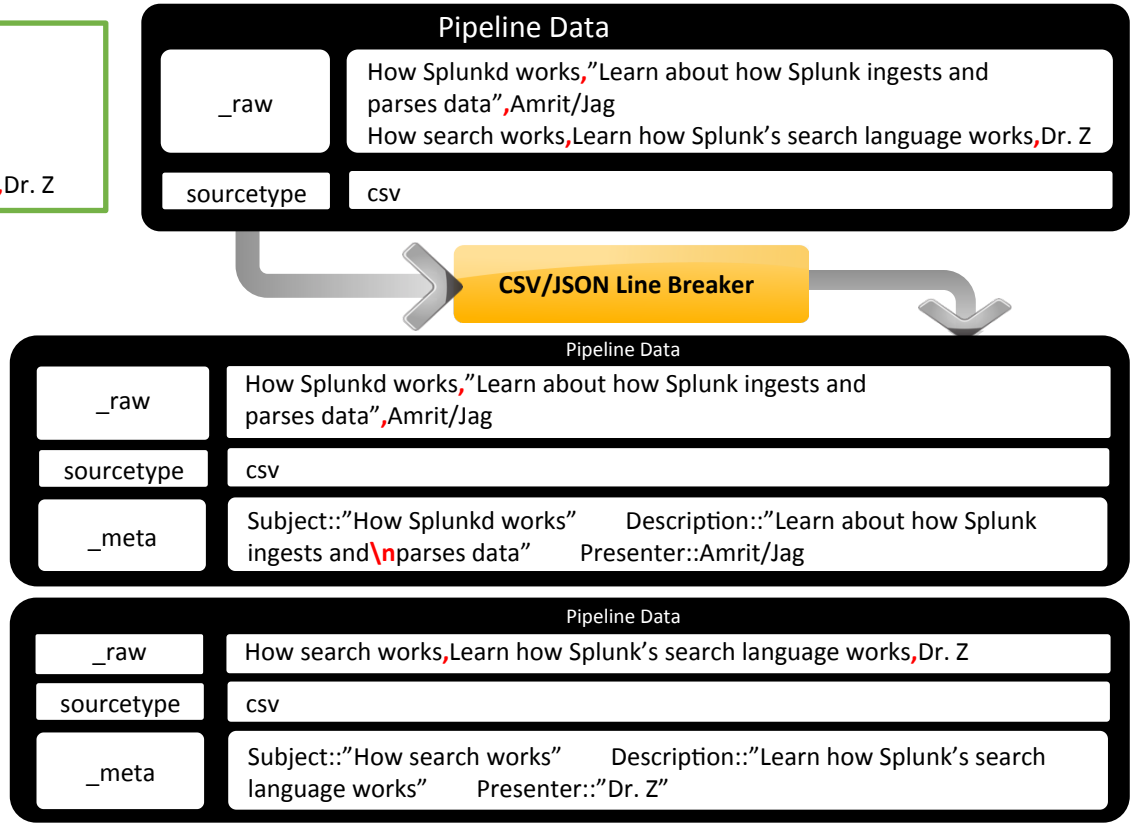
Parsing: Line Breaker



OR Parsing: CSV/JSON Line Breaker (6.0+)

From file containing:

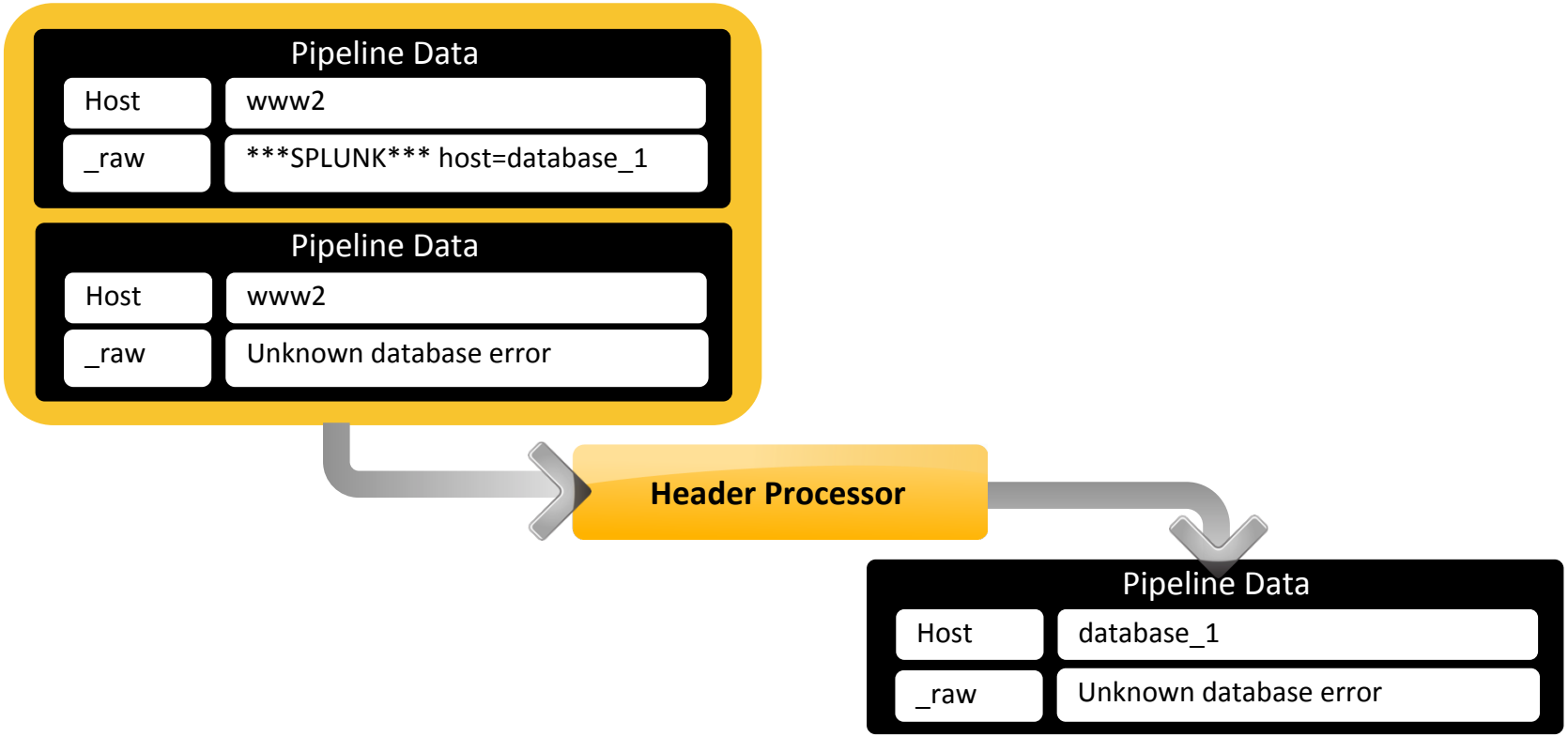
Subject,Description,Presenter ↵
How Splunkd works,"Learn about how Splunk ingests and ↵
parses data",Amrit/Jag ↵
How search works,Learn how Splunk's search language works,Dr. Z



See also:

sourcetype=_json
INDEXED_EXTRACTIONS=(csv|json|…)

Parsing: Header Processor



Merging: Line Merge (aka Aggregator)

Pipeline Data

_raw | Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan

Pipeline Data

_raw | Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: UpdateController: Message tracing {

Pipeline Data

_raw | "interval_since_last_invocation" = 23000;

Pipeline Data

_raw | "power_source" = ac;

Pipeline Data

_raw | Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan

Pipeline Data

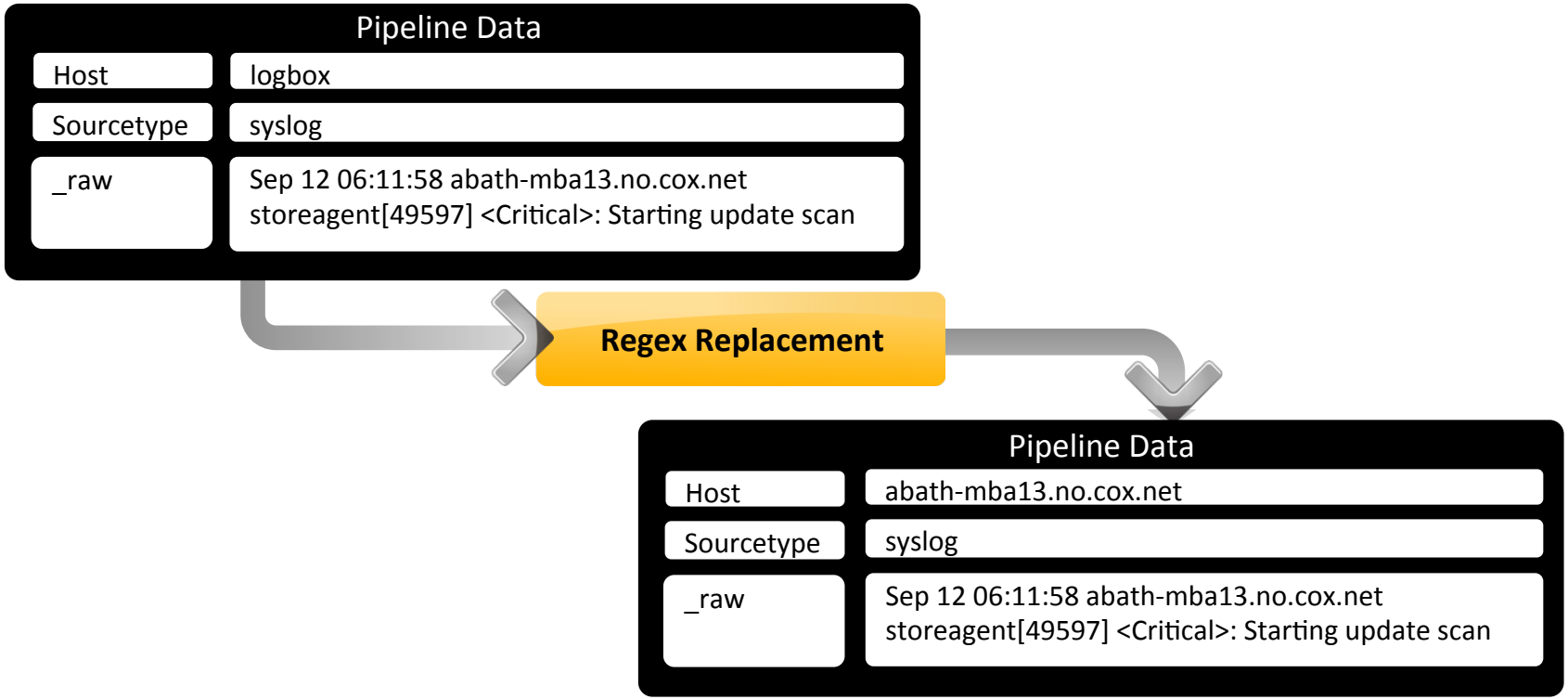
_raw

```
Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: UpdateController: Message tracing {
        "interval_since_last_invocation" = 23000;
        "power_source" = ac;
        "power_state" = wake;
        "start_date" = "2014-08-21 20:10:39 +0000";
}
```

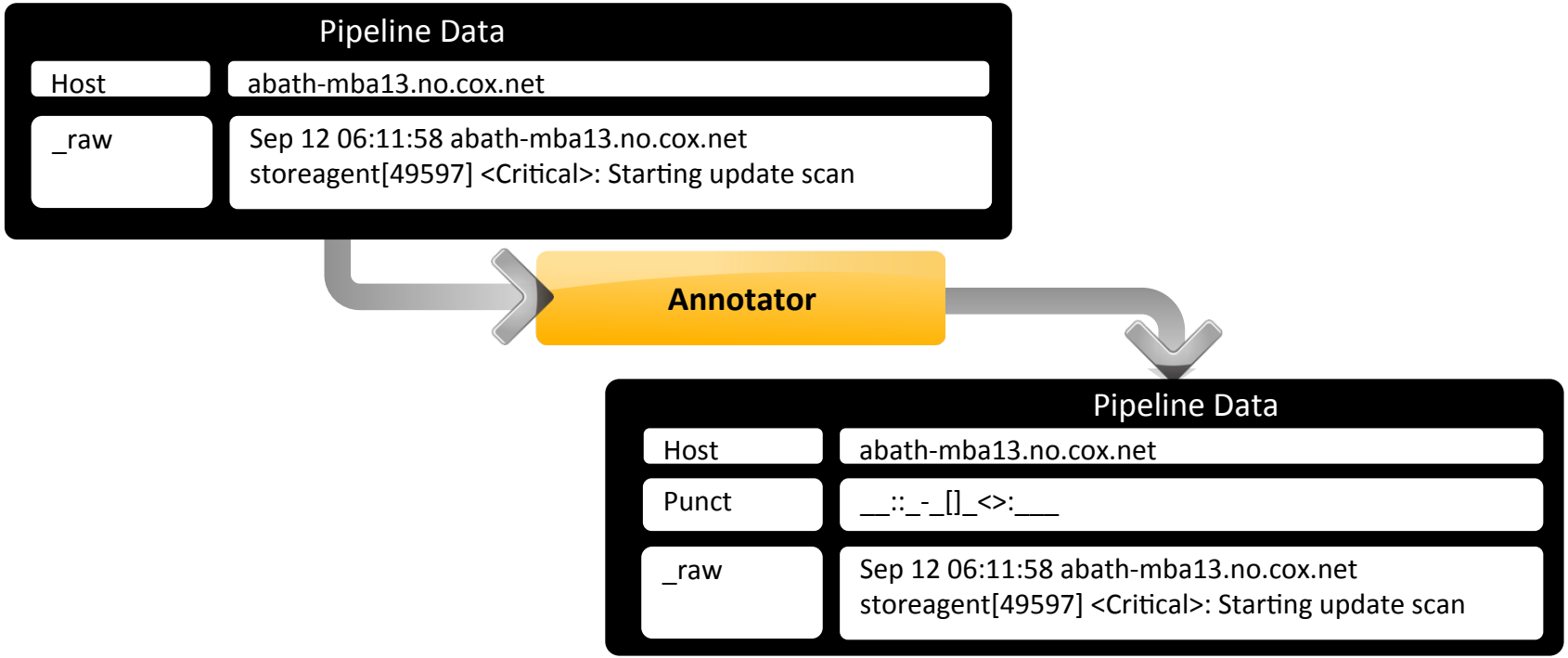
Pipeline Data

_raw | Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Asserted BackgroundTask power assertion (returned 0)

Typing: Regex Replacement

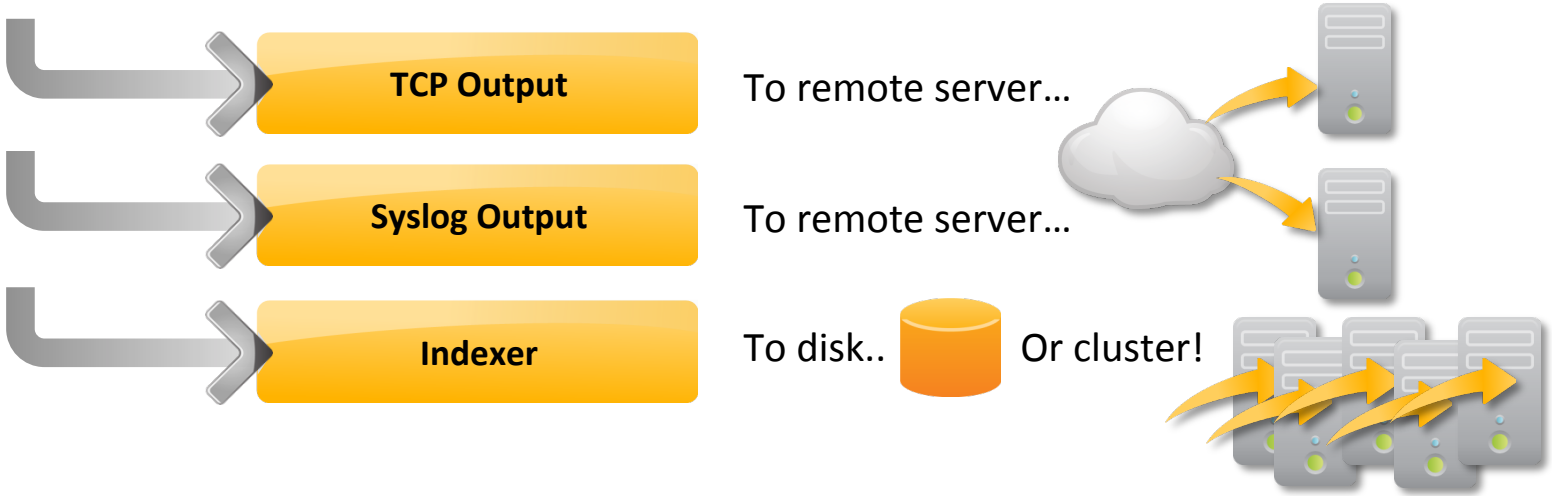


Typing: Annotator

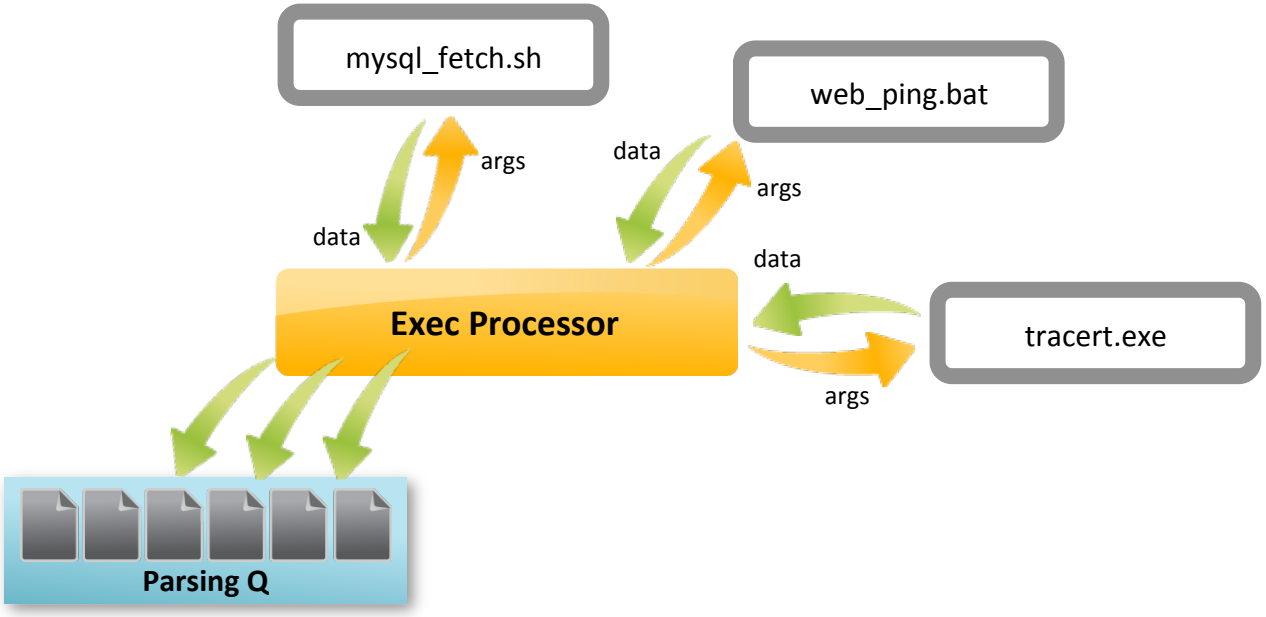


Indexer Pipeline: TCP/Syslog Out, Indexer

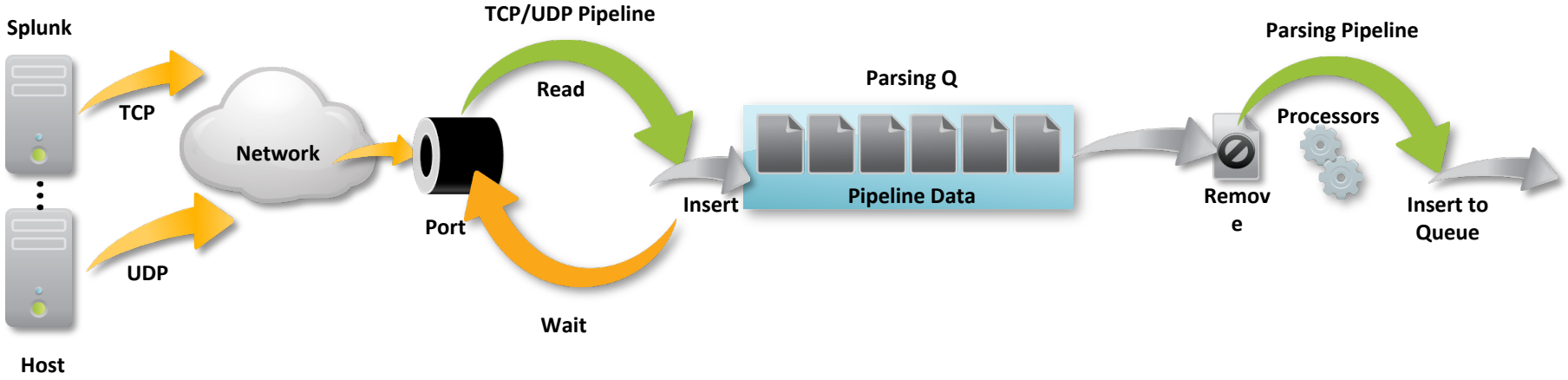
Pipeline Data	
Host	abath-mba13.no.cox.net
Index	main
_raw	Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] <Critical>: Starting update scan



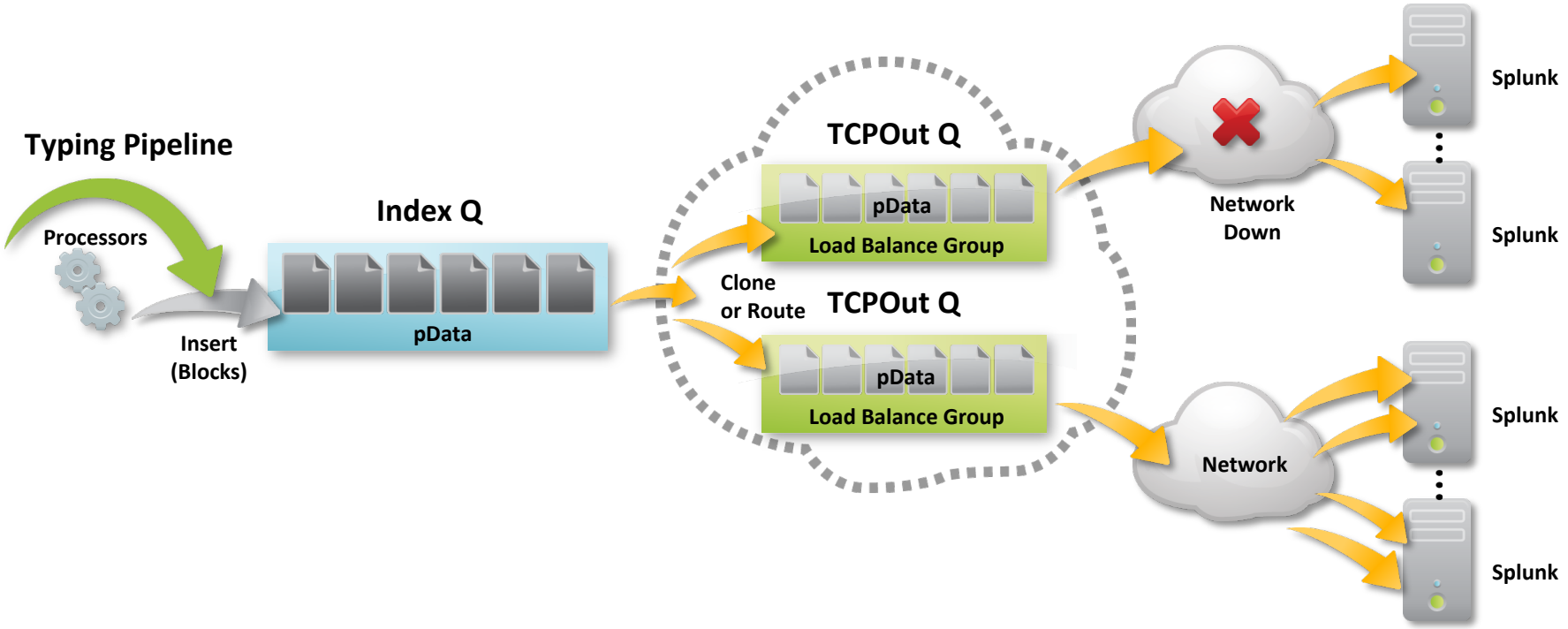
Scripted Input (aka Exec Processor)



TCP/UDP Input



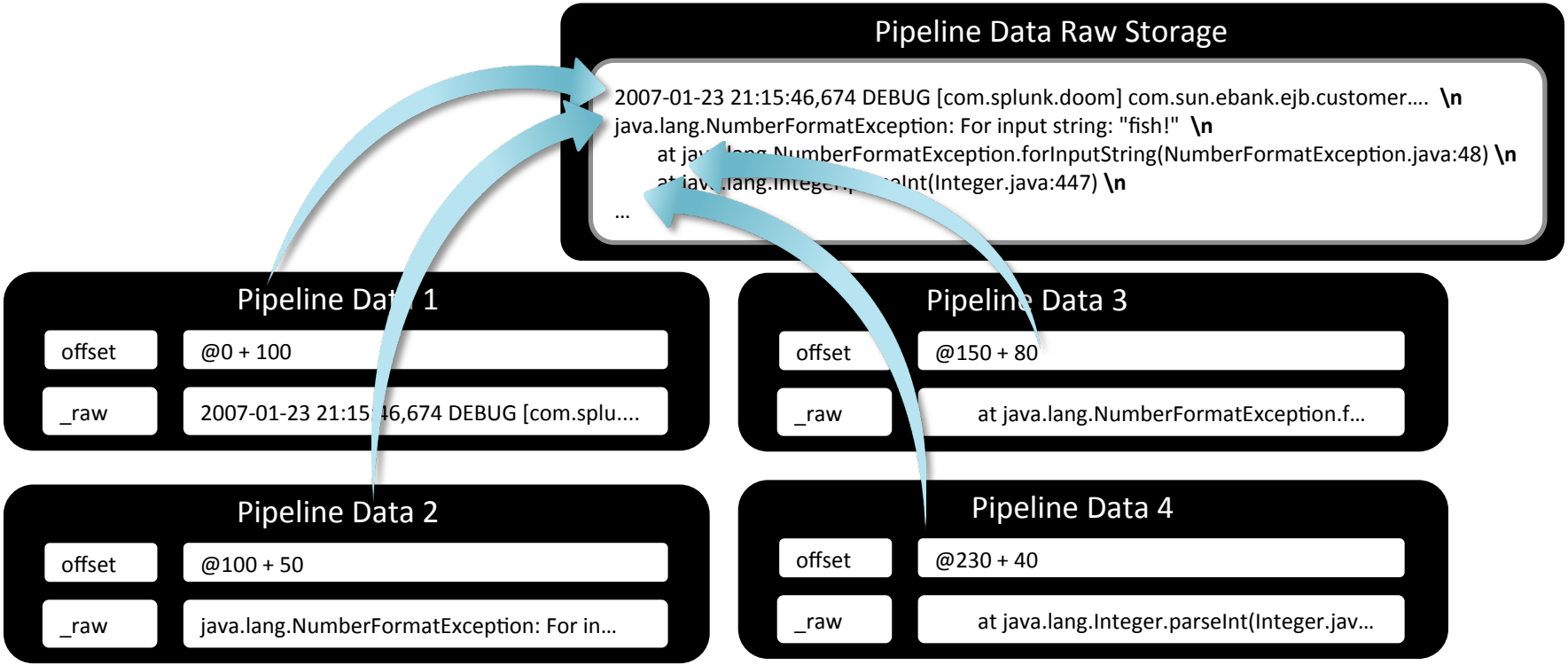
TCP Output Qs



Nerdier Stuff

splunk>

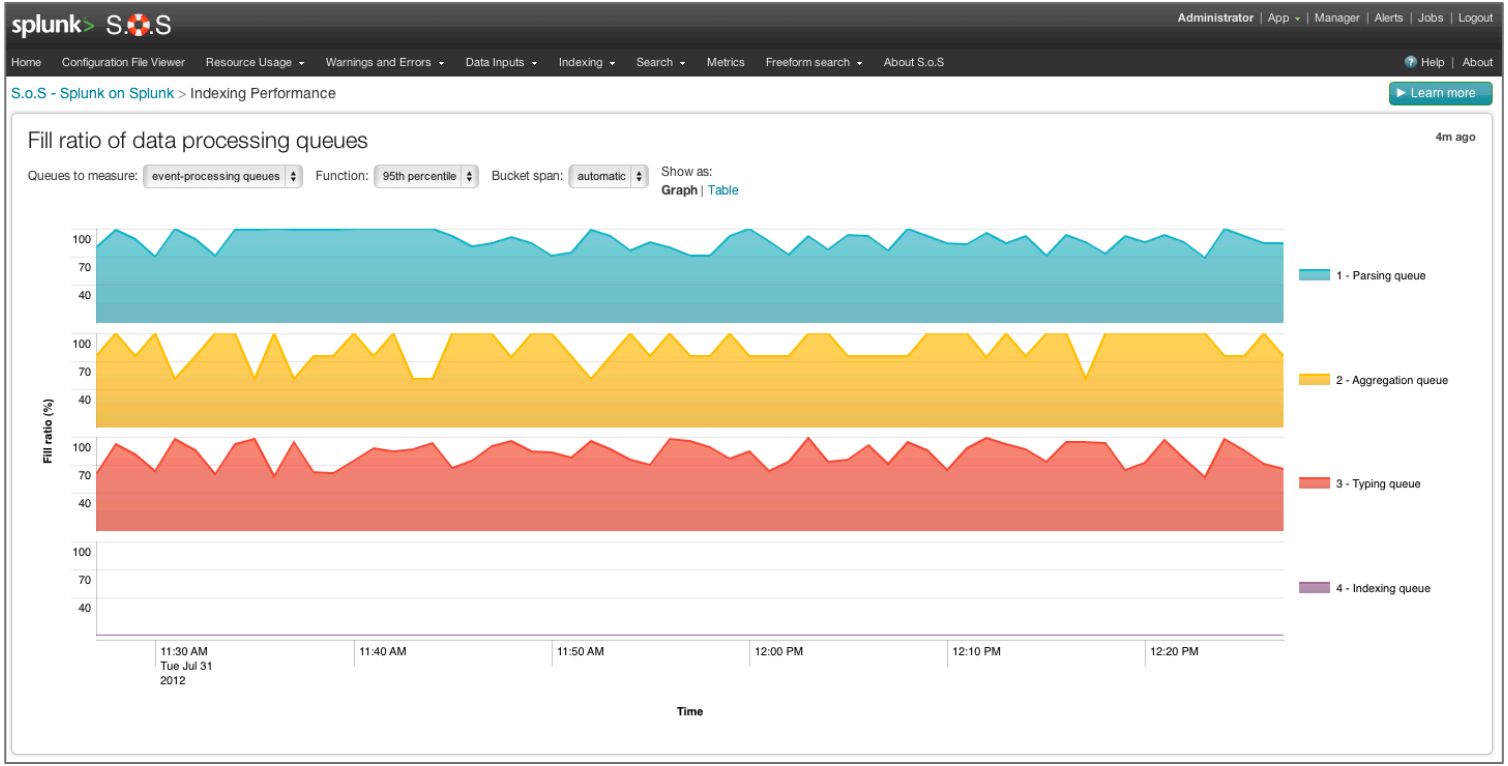
Resource Management



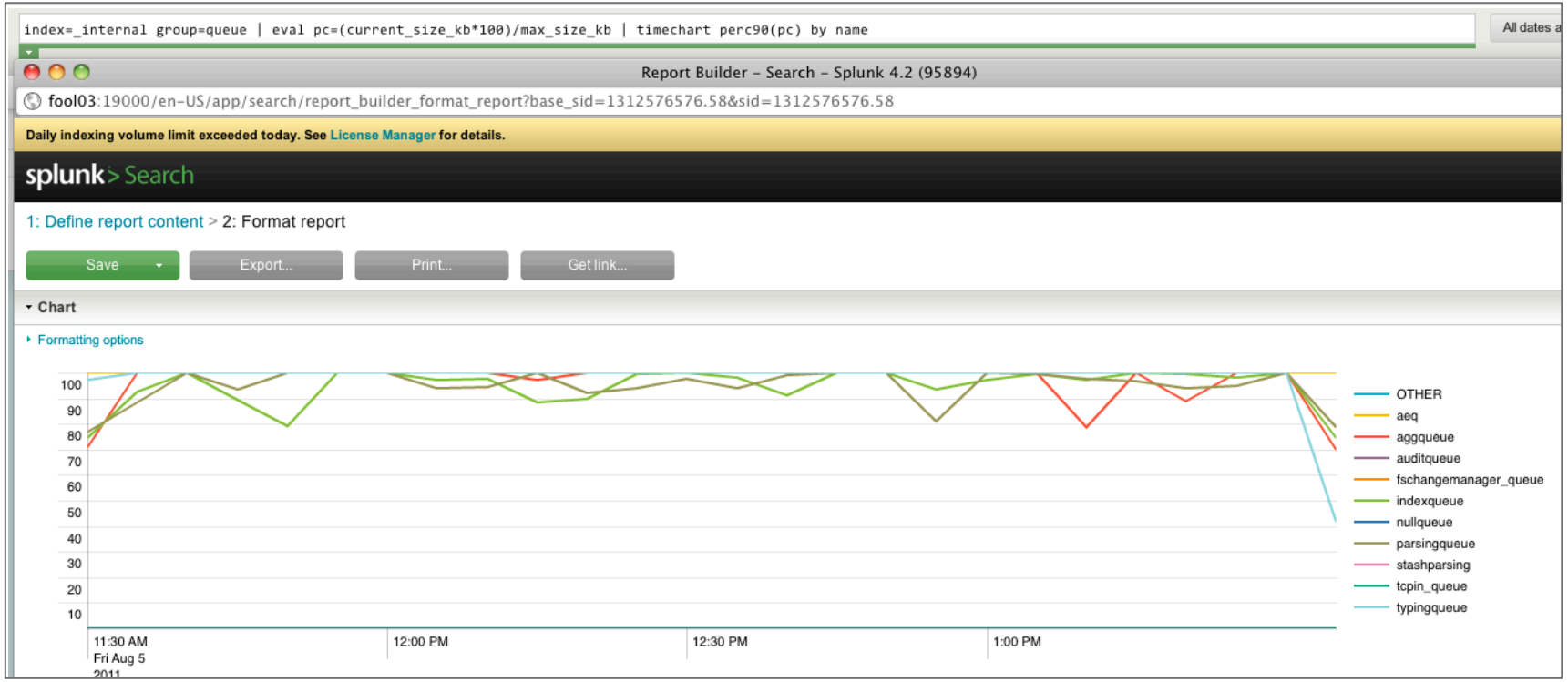
Debugging!
Metrics!
S.O.S App
(and DMC…)!

splunk>

metrics.log: Queues via S.O.S

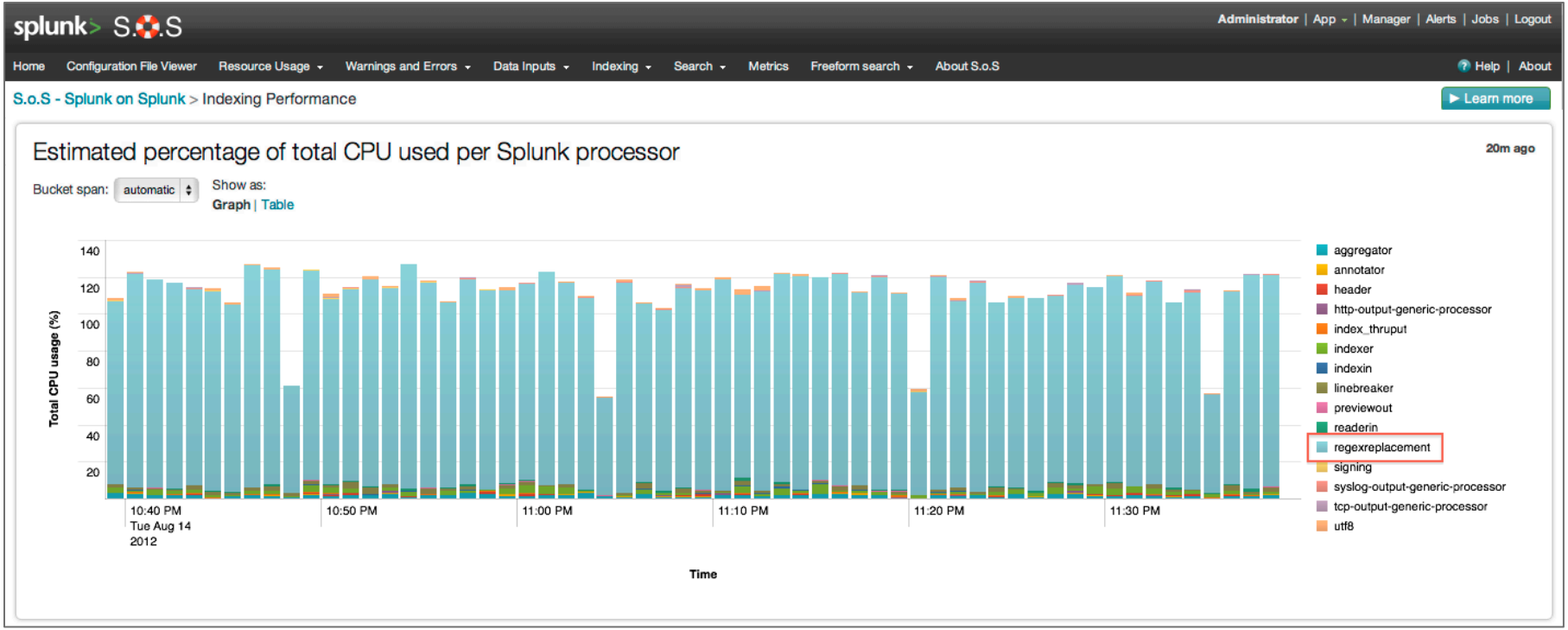


metrics.log: Queues via Search

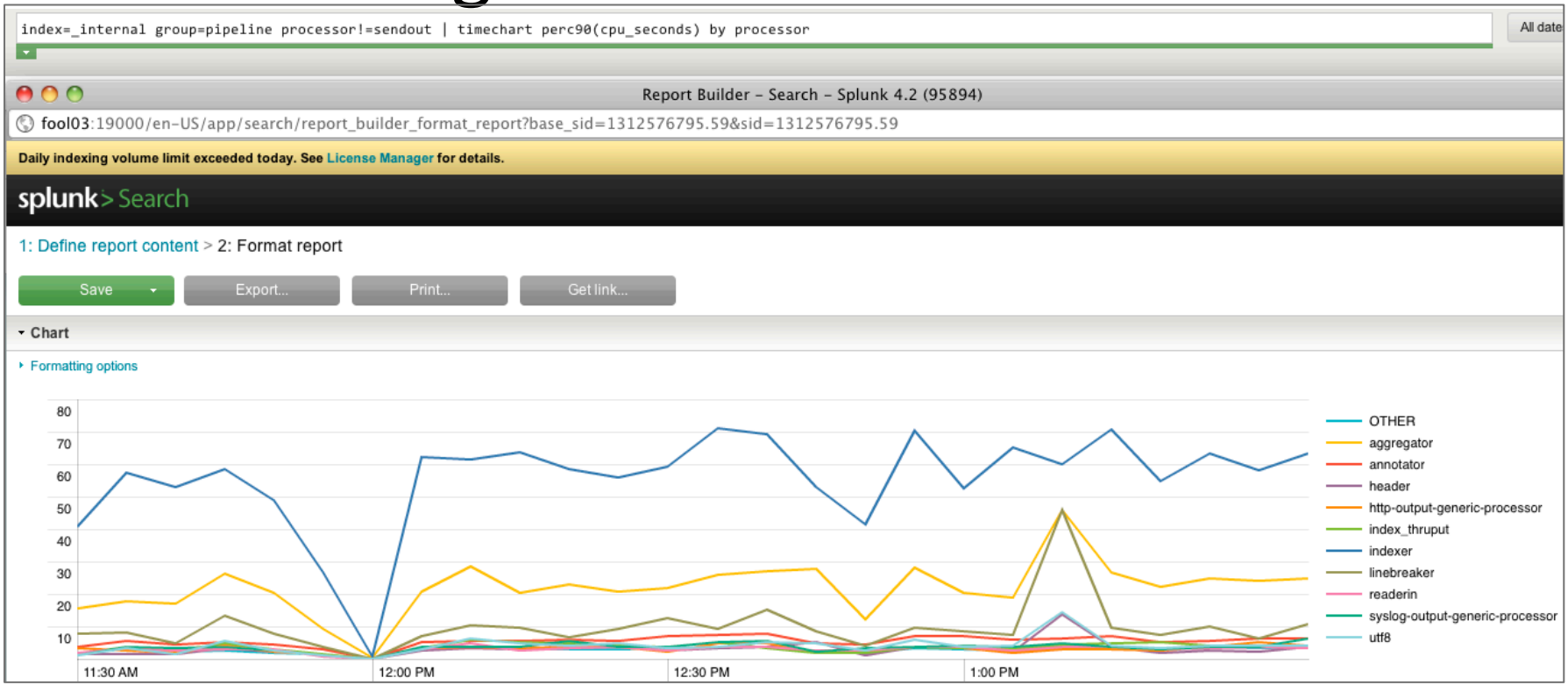


Search: index=_internal group=queue | eval pc=(current_size_kb*100)/max_size_kb | timechart perc90(pc) by name

metrics.log: Processor Time via S.O.S

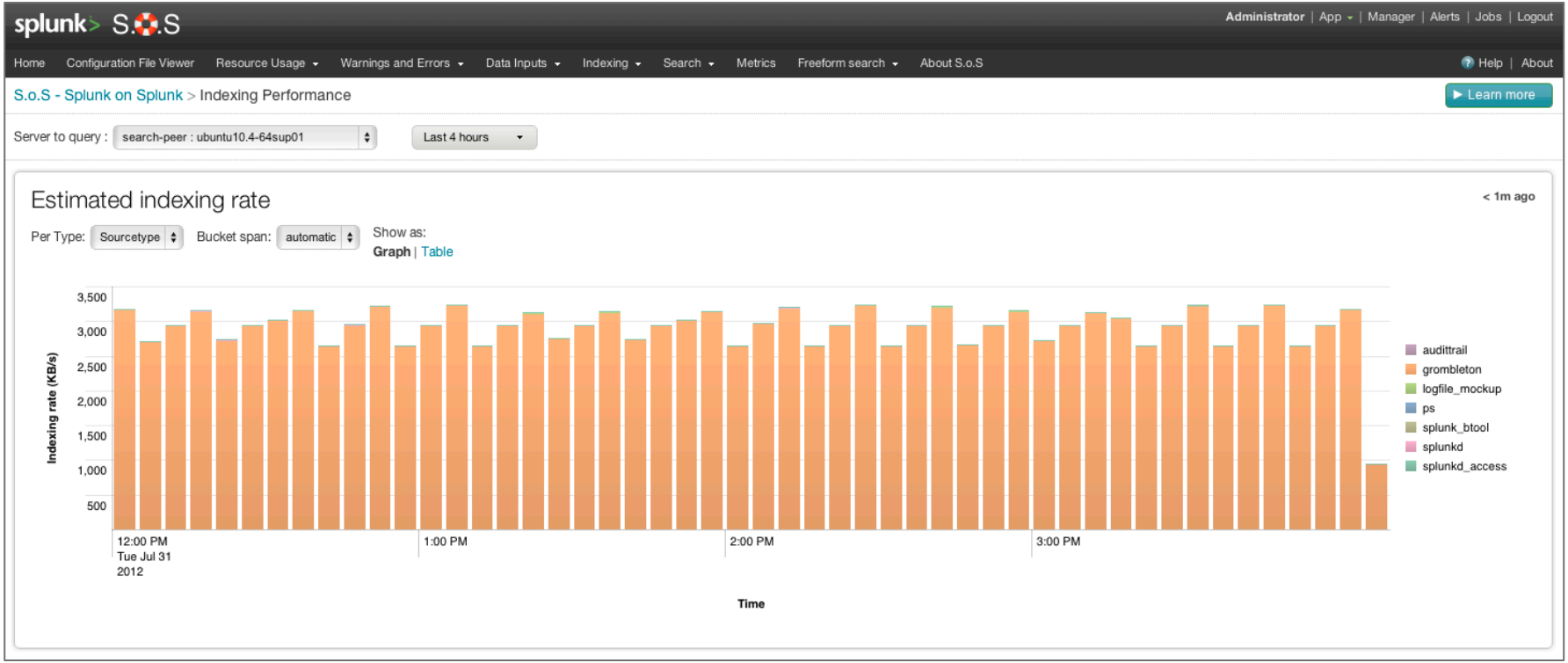


metrics.log: Processor Time via Search

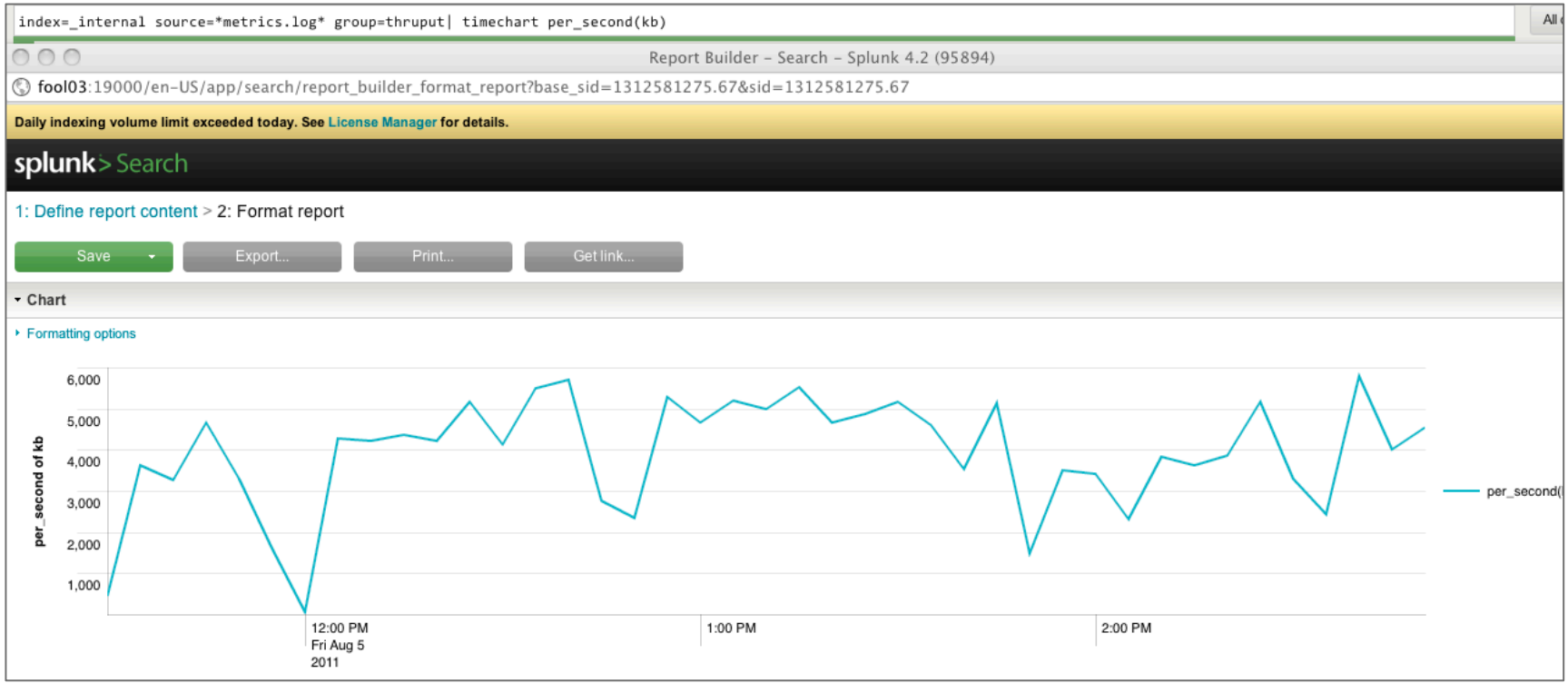


Search: index = _internal group = pipeline processor != sendout | timechart perc90(cpu_seconds) by processor

metrics.log: Indexing Rate via S.O.S



metrics.log: Indexing Rate via Search



Search: index=_internal source=*metrics.log* group=thruput | timechart per_second(kb)

metrics.log: Scenarios

- Indexing instance: Index Queue at 100%
  - Forwarding disabled: Indexing rate? Slow disk? Full disk?
  - Forwarding enabled:
    Indexing rate? Slow disk on remote indexer? Full remote disk?
    TCPOut rate? Low network bandwidth? High network latency?
    Local indexing rate? Slow local disk? Full local disk?

- Universal Forwarder: Parsing Queue at 100%
  - Indexing rate? Slow disk on remote indexer? Full remote disk?
    TCPOut rate? Low bandwidth? High latency?
    (No local indexing here)

- Start from end, work backwards…

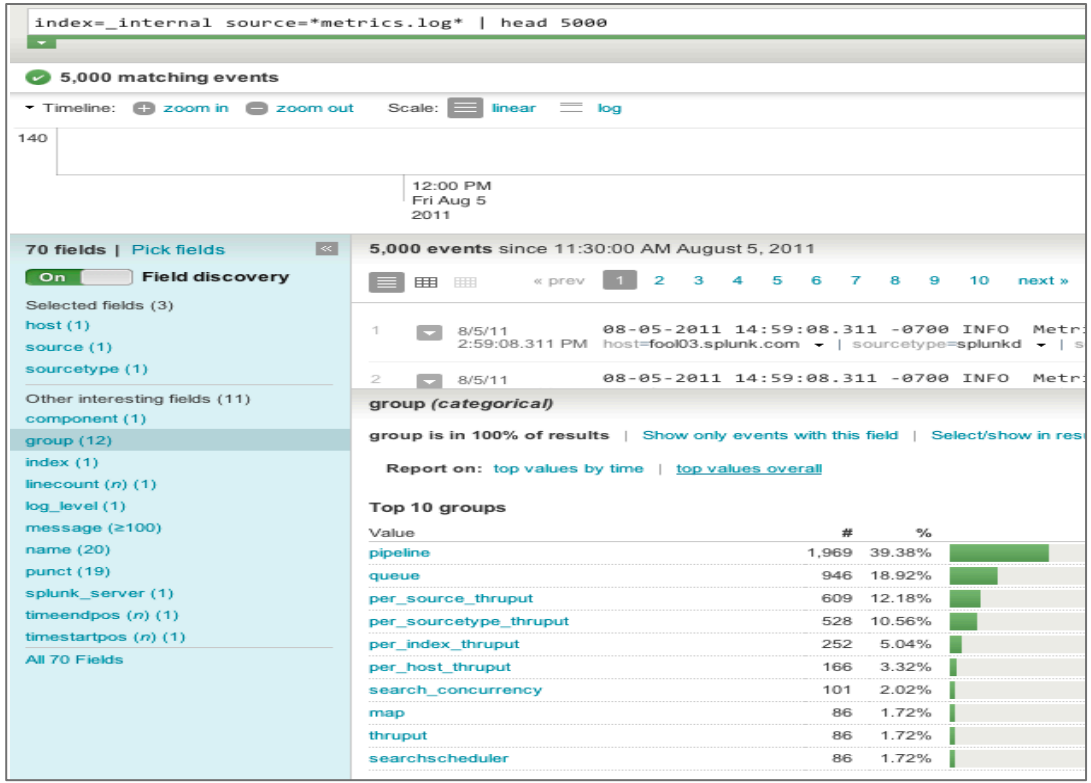
metrics.log: Universal Forwarder

- No indexing/searching capability
- Can forward metrics to indexer…
  - May not get there!
    - Configure S.O.S: http://splunk-base.splunk.com/answers/48874/how-can-i-monitor-the-resource-usage-of-my-forwarder-using-the-sos-app#50315
    - Fix forwarding: http://splunk-base.splunk.com/answers/38091/best-practices-to-deploy-the-splunk-on-splunk-app-in-a-distributed-search-environment
- Fallback to raw file (grep!)

```
$ grep group=queue metrics.log | grep --color 'max_size.*current_size_kb[^,]*,'
Metrics - group=queue, name=typingqueue, blocked=true, max_size_kb=500,
current_size_kb=499, current_size=1821, largest_size=1821, smallest_size=0
```

Metrics Log

- Search:
  index=_internal
  source=*metrics.log*
- Groups
  - pipeline
  - queue
  - per_source_thruput
  - per_sourcetype_thruput
  - per_index_thruput
  - per_host_thruput
  - …



Recap

- Splunk instance consists of linear pipelines
- Splunk topology emulates pipelines
- Downstream slowdown results in upstream blockage
- metrics.log across the topology reveals the whole picture
  - Queue sizes
  - Indexing thruput
  - Forwarding thruput
  - CPU usage per PipelineData Processor
- This is how you should debug – the same way we do!

…

- See also: PipelineSets Talk(Abhinav, Sourav, Tameem)

Questions?

splunk>

THANK YOU

splunk>