How splunkd works

splunkd: Pipelines, Processors, Queues
Inputs: File, Network, Script, HEC, S2S, ...
Debugging: Metrics, Monitoring Console

by Amrit Bath, Abhinav Nekkanti
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, Splunk>, Listen to Your Data, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2017 Splunk Inc. All rights reserved.
Agenda

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

Visual approximation of Octavio
About Us

► Amrit Bath (2005) – Engineer
  • CLI, Deployment Server, Tailing, REST API, Universal Forwarder, Indexed Extractions, Cloud, SHC, Metrics, …
  • Previously: College
  • Hates working on cars
  • Owns Bud Light pants

► Abhinav Nekkanti (2013) – Engineer
  • Tailing, Pipeline Parallelization, Alerts improvements, Preview improvements, Cloud
  • Previously: Citrix (Goto products)
  • Size 14 shoe
  • Overly enthusiastic about Bud Light tallboys
It’s all a pipeline
It’s all a pipeline

60 barrels/ day

A

60 barrels/ day

B

60 barrels/ day

C

60 barrels/ day

D

60 barrels/ day

E

60 barrels/ day

A

B

C

D

E

splunk>
It's all a pipeline

A: 60 barrels/day
B: 60 barrels/day
C: 60 barrels/day
D: 40 barrels/day
E: 40 barrels/day
It’s all a [smaller] pipeline
Really

7200 RPM? Syslog? Log4J?

Forwarder

1 GbE? 100Mb? 10Mb?
Internally, too!

- Monitor Input
- Processors
- Queue
- Queue
- Index

A → B → C → D → E
And across multiple instances!

(This is the biggun’)

Universal Forwarder → Processors

Queue

TCP input

TCP Output

Indexer

Monitor Input

Queue

100 Mb

Index
Data structures & Routing
## Pipeline Data

<table>
<thead>
<tr>
<th>_conf</th>
<th>www2, access_log, /var/log/httpd/access_log</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>www2</td>
</tr>
<tr>
<td>Index</td>
<td>prod_servers</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>_raw</td>
<td>10.3.1.92 - - [21/Jul/2011:20:34:44 -0700] &quot;GET /results/bonnie-solns_vm_nick.html HTTP/1.1&quot; 200 2938</td>
</tr>
</tbody>
</table>

- UTF-8 finished?
- Line Breaker finished?
-...
-...
Processor

- Processor: Performs small but logical unit of work
- Contained within a Pipeline
- Examples: LineBreaker, Aggregator, TcpInput, Index
Pipelines

- Each pipeline runs in separate thread
- Naturally parallelized and modular
- Data flows linearly
- Configured in: $SPLUNK_HOME/etc/modules/
- Rendered to: $SPLUNK_HOME/var/run/splunk/composite.xml
Complete Ingestion Pipeline Set

- **Parsing Pipeline**
  - utf8
  - linebreaker
  - header

- **Merging Pipeline**
  - aggregator

- **Typing Pipeline**
  - regex replacement
  - annotator

- **Index Pipeline**
  - tcp out
  - syslog out
  - indexer

- **UF**
  - TCP/UDP
  - Tailing
  - Modular Input
  - HTTP Event Collector
  - Scripted Input
Queue

- Queue size bounded by memory
- Holds variable sized Pipeline Data
Queue

Pipeline

Typing Pipeline

1. Process
2. Insert (Blocks)

Index Queue (Full)

pData pData pData pData

Index Pipeline

Tcp Output

Processors

1. Remove
2. Write to Network (Fails)

Network Down

Processors (Fails)

Processors (Full)

Processors (Blocks)
Persistent Queue

Internal Queues Full

Input Q  pData  pData  Tcpout Q

Persistent Q  Full

Much Bigger Queue

Network

Host

Splunk

Network

RAM

File System
Input Processors
Input Pipelines

Pipeline Data

- **Host**: amritDesktop
- **Index**: prod_servers

_RAW_

Jul 30 00:21:19 amritDesktop sshd[30416]: Accepted publickey for amrit from 10.3.1.52 port 59426 ssh2

Jul 30 00:21:26 amritDesktop sshd[30418]: Received disconnect from 10.3.1.52: 11: disconnected by user
Files with the same contents are indexed only once (initCrc)

64KB PipelineData created per read

Files are read serially (per reader)

All readers are per-pipelineset
Scripted Input

- mysql_fetch.sh
- web_ping.bat
- tracert.exe
1) Retrieve a configuration scheme

2) Generate data

---

<scheme>
<title>S3 Input</title>
<description>
Index log files from AWS S3.
....
</description>
</scheme>

<stream>
  <event>
    <data> File 1 contents </data>
    <source> s3:/svc/Foo.gz </source>
    <time> 1474483334 </time>
  </event>
</stream>
HTTP Event Collector

```
{"time": 1426279439, // epoch time
"host": "rasp_pi_270b",
"source": "temp_kegerator_4",
"sourcetype": "temp_sensor",
"index": "beer",
"event": { "keg_f=38 ambient_f=70" }
}```
TCP/UDP Input

TCP/UDP Processor

Read

Wait

Insert

Parsing Q

Pipeline Data

Splunk Host TCP UDP

Network

Port

Read

Insert

TCP/UDP Input

TCP/UDP Processor

Read

Wait

Insert

Parsing Q

Pipeline Data

Splunk Host TCP UDP

Network

Port
Mining Processors

We’re getting there
Parsing: UTF-8 Processor

props.conf:
[my_sourcetype]
CHARSET = Shift-JIS

Pipeline Data

<table>
<thead>
<tr>
<th>Host</th>
<th>www2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charset</td>
<td>Shift-JIS</td>
</tr>
<tr>
<td>_raw</td>
<td>fjl&quot;fBfa\bN Efgf%of\UF€</td>
</tr>
</tbody>
</table>

UTF-8 Processor

Pipeline Data

<table>
<thead>
<tr>
<th>Host</th>
<th>www2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charset</td>
<td>UTF-8</td>
</tr>
<tr>
<td>_raw</td>
<td>ポンティアク・トランザム</td>
</tr>
</tbody>
</table>
Sep 12 06:11:58 abath-mba.no.cox.net storeagent[49597] <Critical>: Starting update scan

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

Sep 12 06:11:58 abath-mba.no.cox.net storeagent[49597] <Critical>: Asserted BackgroundTask power
How Splunkd works,
"Learn about how Splunk ingests and parses data",
Amrit/Abhinav

How search works,
Learn how Splunk’s search language works,
Dr. Z

inputs.conf:
sourcetype = _json
props.conf:
INDEXED_EXTRACTIONS = (csv | json | …)
**Parsing: Header Processor**

**Pipeline Data**
- **Host**: www2
- **_raw**: ***SPLUNK*** host=database_1

**Pipeline Data**
- **Host**: www2
- **_raw**: Unknown database error

**props.conf**: HEADER_MODE = (none | firstline | always)

**Pipeline Data**
- **Host**: database_1
- **_raw**: Unknown database error
### Typing: Regex Replacement

#### Pipeline Data

<table>
<thead>
<tr>
<th>Host</th>
<th>logbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sourcetype</td>
<td>syslog</td>
</tr>
<tr>
<td>_raw</td>
<td>Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] &lt;Critical&gt;: Starting update scan</td>
</tr>
</tbody>
</table>

#### Regex Replacement

```
props.conf:
[syslog]
TRANSFORMS = syslog-host

transforms.conf:
[syslog-host]
DEST_KEY = MetaData:Host
FORMAT = host:$1
REGEX = . . . ( . . . ) . . .
```

#### Pipeline Data

<table>
<thead>
<tr>
<th>Host</th>
<th>abath-mba13.no.cox.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sourcetype</td>
<td>syslog</td>
</tr>
<tr>
<td>_raw</td>
<td>Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] &lt;Critical&gt;: Starting update scan</td>
</tr>
</tbody>
</table>
## Typing: Annotator

### Pipeline Data

<table>
<thead>
<tr>
<th>Host</th>
<th>abath-mba13.no.cox.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>_raw</td>
<td>Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] &lt;Critical&gt;: Starting update scan</td>
</tr>
</tbody>
</table>

### Annotator

- **props.conf:**
  - ANNOTATE_PUNCT = (true | false)

### Pipeline Data

<table>
<thead>
<tr>
<th>Host</th>
<th>abath-mba13.no.cox.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punct</td>
<td>___:: _: - [] &lt;&gt;: ___</td>
</tr>
<tr>
<td>_raw</td>
<td>Sep 12 06:11:58 abath-mba13.no.cox.net storeagent[49597] &lt;Critical&gt;: Starting update scan</td>
</tr>
</tbody>
</table>
Index & Output Processors

We’re still here…
Indexer Pipeline: Fwding, Indexing

<table>
<thead>
<tr>
<th>Pipeline Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
</tr>
<tr>
<td><strong>Index</strong></td>
</tr>
<tr>
<td><strong>_raw</strong></td>
</tr>
</tbody>
</table>

**TCP Output**
To remote server

**Syslog Output**
To remote server

**Indexer**
To disk or cluster
TCP Output Qs

Typing Pipeline

Processors

Insert (Blocks)

Index Q

TCPOut Q

pData

Load Balance Group

TCPOut Q

pData

Load Balance Group

Network

Splunk

Splunk

Splunk

Splunk

Network
Forwarding and Receiving

Hi
Forwarding with ACK

Log file → Forwarder → Indexer → Disk

outputs.conf:
[tcpout:foobar]
useACK = true
Multiple Pipeline Sets

More faster, sometimes
Complete Ingestion Pipeline Set (Recap)

- **Parsing Pipeline**:
  - utf8
  - linebreaker
  - header

- **Merging Pipeline**:
  - aggregator

- **Typing Pipeline**:
  - regex replacement
  - annotator

- **Index Pipeline**:
  - tcp out
  - syslog out
  - indexer

**UF**
- TCP/UDP
- Tailing
- Modular Input
- HTTP Event Collector
- Scripted Input
With Multiple Pipeline Sets (6.3+)

- TCP/UDP
- Tailing
- Modular Input
- HTTP Event Collector
- Scripted Input

UF

<table>
<thead>
<tr>
<th>Parsing Pipeline</th>
<th>Merging Pipeline</th>
<th>Typing Pipeline</th>
<th>Index Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pipeline Sets: CPU Utilization

- Rule of Thumb:
  - 4-6 cores per set
  - 1 core per search (unless batch_search_max_pipeline > 1…)

- Example: Indexer host with 2 Pipeline Sets:
  - 2 x 6 = 12 cores for splunkd daemon
  - 10 searches running
  - Other cores: “splunk-optimize”, etc, periodically.
  - Total: Roughly 10 + 12 = 22 cores in use.

- Parallel pipeline sets for underutilized hosts:

```conf
server.conf:
[general]
parallelIngestionPipelines = 2
```
Pipeline Sets: Not an easy decision!

- Can increase forwarding or indexing rate:
  - Total rate = $\text{base\_rate} \times \text{num\_pipeline\_sets}$
  - **Good idea** on edge Forwarders *(see next slide…)*
  - **Bad idea** on already-overwhelmed indexers!

- **Does not reduce search time!**
  - Search time = $\frac{\text{base\_search\_time}}{\text{number\_machines}}$

- Assume: indexing $N$ GB/day
  - On 4 indexers, pipeline sets = 1.
    - Dense search, 24 hours: Each indexer searches $(N / 4)$ GB.
  - Increase to pipeline sets = 2, downsize to 2 indexers.
    - Dense search, 24 hours: Each indexer searches $(N / 2)$ GB.
  - Same indexing rate with fewer machines, but **searches take twice as long!**
    - Except with "batch mode" searches...
Multiple Ingestion Pipeline Sets over Network

Forwarder: 3 Pipeline Sets

Indexer

3 Indexers: 1 Pipeline Set each
Nerdier Stuff

8^-}
java.lang.NumberFormatException: For input string: "fish!"

at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)

at java.lang.Integer.parseInt(Integer.java:447)

...
Debugging! Monitoring Console! (aka DMC!)
Pipelines/Processors (Debugging)
Pipelines/Processors (Debugging)

TCP/UDP pipeline
Tailing
FIFO pipeline
FSChange
Exec pipeline

Pars Ing Queue
Merging Pipeline
Typing Pipeline
Index Pipeline

regex replacement
annotator
tcp out
syslog out
indexer
Pipelines/Processors (Debugging)

TCP/UDP pipeline
Tailing
FIFO pipeline
FSChange
Exec pipeline

Parsing Pipeline
- utf8
- linebreaker
- header

Merging Pipeline
- Agg Queue
- aggregator

Typing Pipeline
- Typing Queue
- regex replacement
- annotator

Index Pipeline
- tcp out
- syslog out
- indexer

Tailing FIFO pipeline
FSChange Exec pipeline
DMC/Monitoring Console – Indexing Performance
DMC/MC – Queues/Pipelines

Splunk Enterprise Data Pipeline

tcp / udp

tail

eexec

Parsing Pipeline

time

Merging Pipeline

Typing Pipeline

Index Pipeline

Disk

syslog out

tcp out

Index Queue

Punctuation extraction

regexp-replacement

annotator

Fill Ratio

94%

Fill Ratio

81%

Line merging

Time extraction

Fill Ratio

83%

header

utf8

linebreaker

Parsing Queue

Aggregator Queue

Fill Ratio

Tail

exec
DMC/MC – CPU per Processor (Unhealthy)
DMC/MC – Indexing Rate

Estimated Indexing Rate Per Sourcetype

Split by
Sourcetype

Indexing Rate (KB/s)

<table>
<thead>
<tr>
<th>Time</th>
<th>Indexing Rate</th>
<th>Sourcetype</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:15 PM</td>
<td>5,000</td>
<td>audittrail</td>
</tr>
<tr>
<td>4:20 PM</td>
<td>5,000</td>
<td>kvstore</td>
</tr>
<tr>
<td>4:25 PM</td>
<td>5,000</td>
<td>mongod</td>
</tr>
<tr>
<td>4:15 PM</td>
<td>5,000</td>
<td>scheduler</td>
</tr>
<tr>
<td>4:20 PM</td>
<td>5,000</td>
<td>splunk_disk_objects</td>
</tr>
<tr>
<td>4:25 PM</td>
<td>5,000</td>
<td>splunk_resource_usage</td>
</tr>
<tr>
<td>4:15 PM</td>
<td>5,000</td>
<td>splunk_web_service</td>
</tr>
<tr>
<td>4:20 PM</td>
<td>5,000</td>
<td>splunkd</td>
</tr>
<tr>
<td>4:25 PM</td>
<td>5,000</td>
<td>splunkd_access</td>
</tr>
<tr>
<td>4:15 PM</td>
<td>5,000</td>
<td>splunkd_ui_access</td>
</tr>
<tr>
<td>4:20 PM</td>
<td>5,000</td>
<td>syslog</td>
</tr>
</tbody>
</table>
DMC/MC - CPU per Processor (Healthy)
metrics.log

- Search:
  - index=_internal
  - source=metrics.log

- Groups
  - pipeline
  - queue
  - thruput
  - per_source_thruput
  - per_sourcetype_thruput
  - per_index_thruput
  - per_host_thruput
  - tcpin_connections
  - tcpout_connections
  - search_concurrency
  - clusterout_connections
  - deploy-server
  - tailingprocessor
metrics.log: SPL searches

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

- **CPU Usage by Splunk Processor:**
  - `index = _internal group=pipeline processor!=sendout`
  - `| timechart perc90(cpu_seconds) by processor`

- **Throughput:**
  - `index=_internal source=*metrics.log* group=thruput`
  - `| timechart per_second(kb)`
metrics.log: Scenarios

- Indexing instance: Index Queue at 100%
  - Forwarding disabled:
    ‣ Indexing rate: High? Lots of data. Low? Check cpu_seconds, iowait, etc. Zero? Disk full!
  - Forwarding enabled: See above, on downstream indexers.
  - Forwarding enabled, and downstream indexers are fine?
    ‣ TCPOut rate: High? Lots of data. Low? See outputs.conf maxKBps, check bandwidth & latency…
      Zero? Check SSL, ports, splunkd.log…
    ‣ Also indexing locally? See “Forwarding disabled” above…

- Universal Forwarder
  - Similar, but: Parsing Queue instead of Index Queue.

- Start from end (healthiest node), work backwards…
metrics.log: Universal Forwarder

- No indexing/searching capability
- Can’t forward logs to indexers if forwarding is busted
- Fallback to raw file (grep!)

1) $ 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

2) $ grep group=tcpout_connections metrics.log
   ... destlp=10.159.4.67, destPort=9997 ... _tcp_KBps=27674.87 ... _tcp_Kprocessed=802571

3) $ grep -E 'ERROR|WARN|FATAL|CRIT' splunkd.log

4) $ splunk help list inputstatus

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 throughput
  - Forwarding throughput
  - CPU usage per PipelineData Processor
- This is how you should debug – the same way we do!
Questions?