Harnessing 6.3 Performance and Scalability

Abhinav Nekkanti
Tameem Anwar
Sourav Pal
Splunk

splunk>

Disclaimer

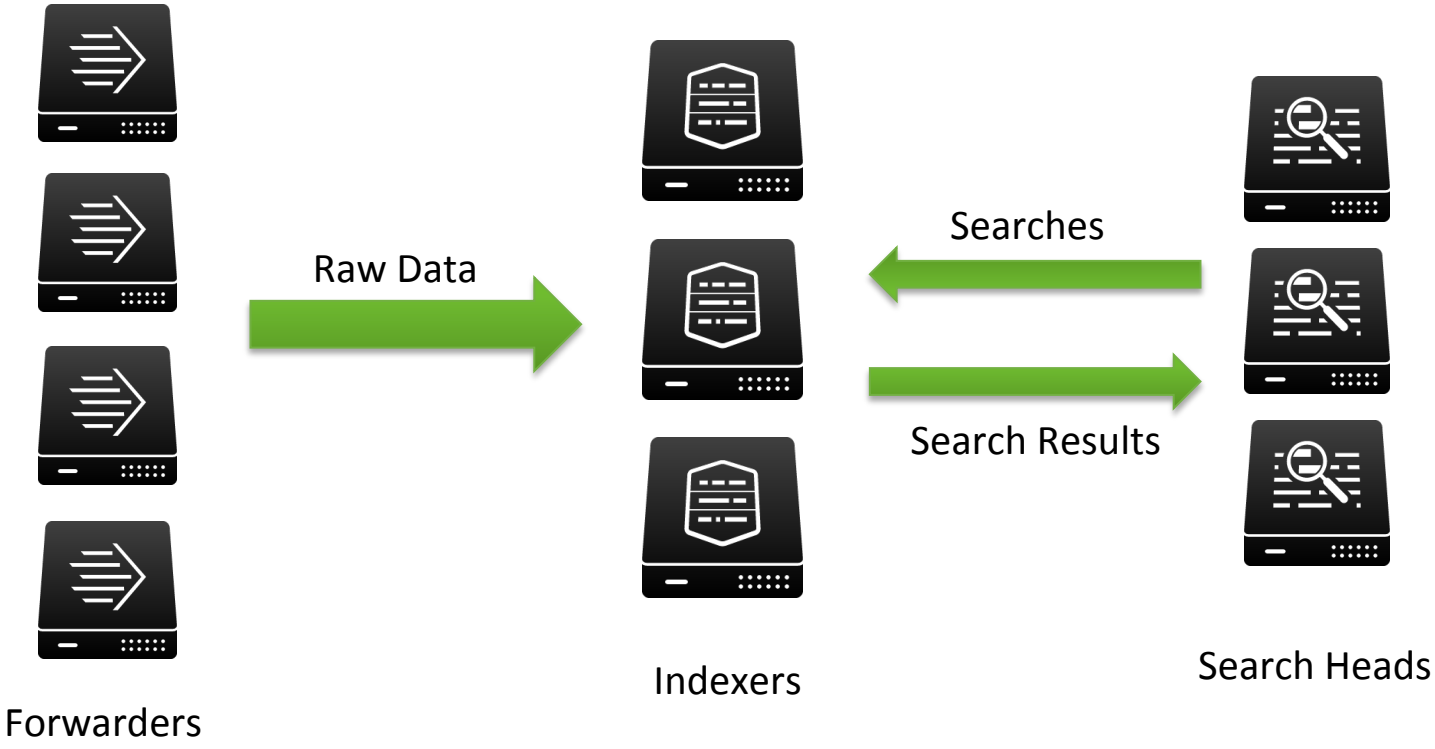
During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results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

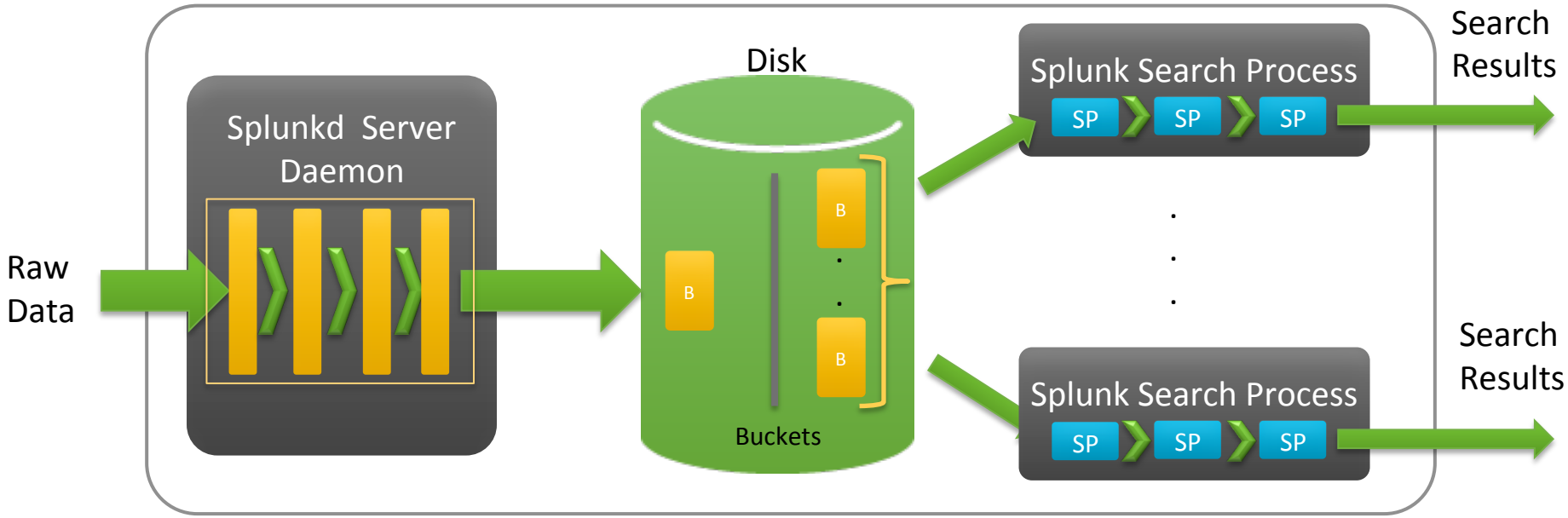
About Us

- Abhinav Nekkanti - Sr. Software Engineer, Splunk
- Sourav Pal - Sr. Software Engineer, Splunk
- Tameem Anwar - Software Engineer - Performance, Splunk

3 Tier Architecture

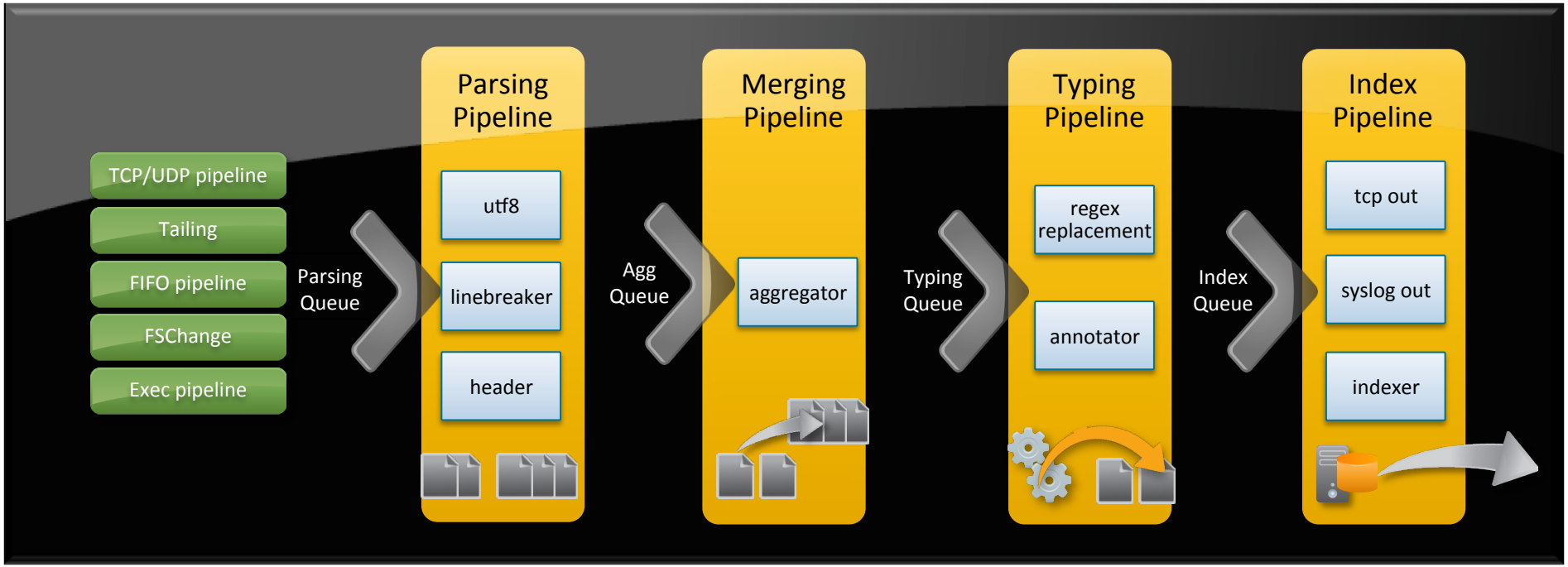


Insight into the Indexer



Traditional Indexer Hosts

Splunkd Server Daemon / Pipelineset



Ingestion Pipeline Set

Indexer Core Utilization

- Rule of Thumb:

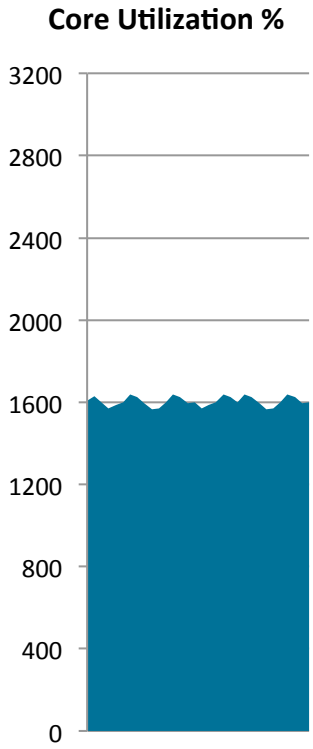
Process	Cores (approx.)
Splunkd Server Daemon	4 to 6 cores
Splunk Search Process	1 core / search process

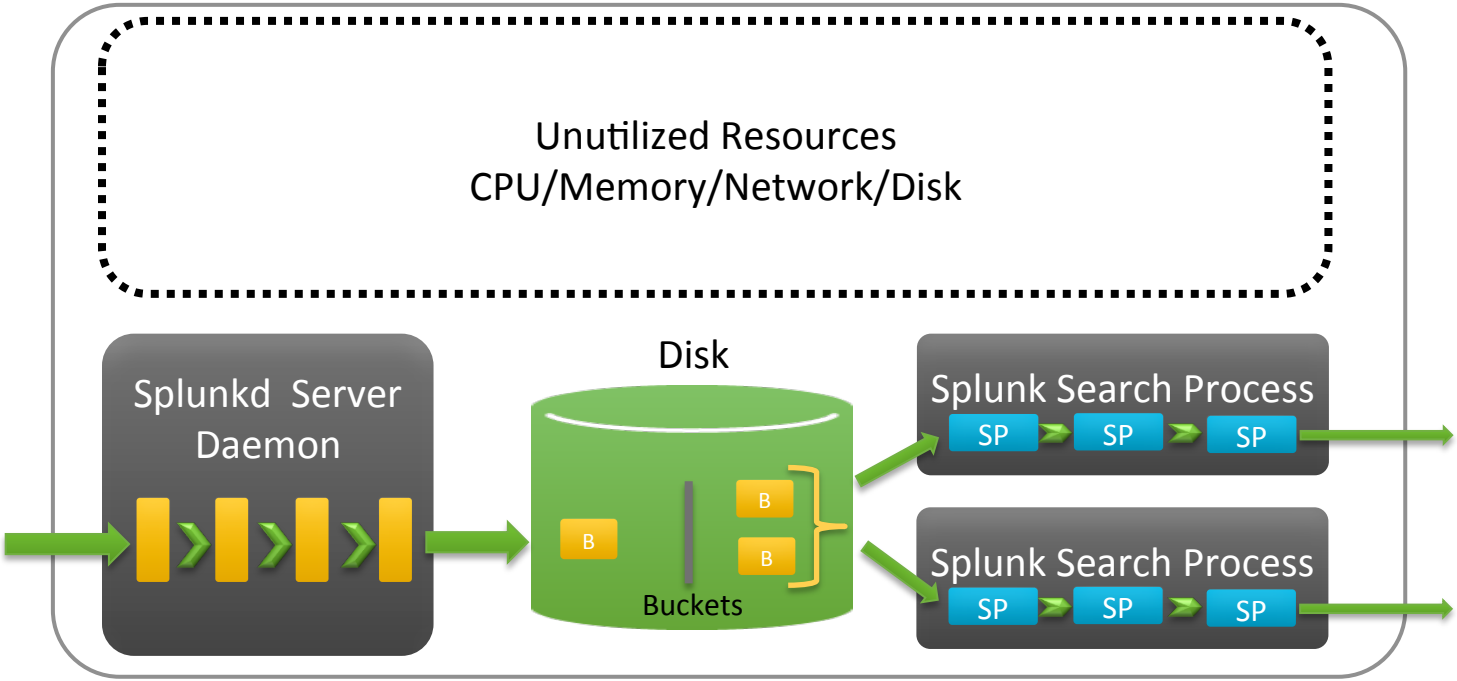
- Example core utilization of a Indexer Host:
  - 4 To 6 cores for Splunkd Server daemon
  - 10 X 1 Cores for Splunk Search Processes
  - Total cores used: 14 to 16 cores

Today's Commodity Hardware

- Dell PowerEdge R820
- Intel(R) Xeon(R) CPU E5-4620 0 @ 2.20GHz - 32 cores
- 8 x 146Gb SAS 15k disks
- 128GB RAM

Under-Utilized Indexer





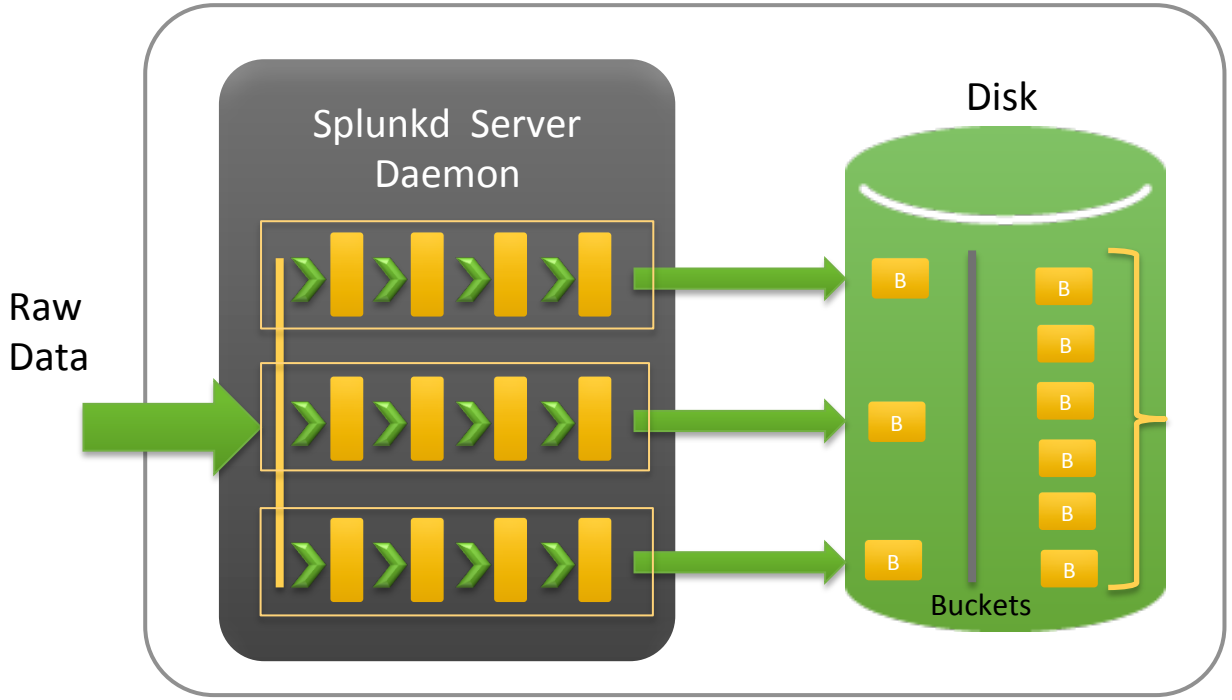
Performance Enhancements in 6.3

- Multiple Pipeline Sets
  - Parallel ingesting pipeline sets
  - Improves resource utilization of the host machine
- Search Improvements
  - Faster batch searches using parallel search pipelines

Multiple Ingestion Pipeline Sets

splunk>

Splunkd with Multiple Ingestion Pipeline Sets



Indexer with 3 Pipeline Sets

Configuring Multiple Ingestion Pipeline Sets

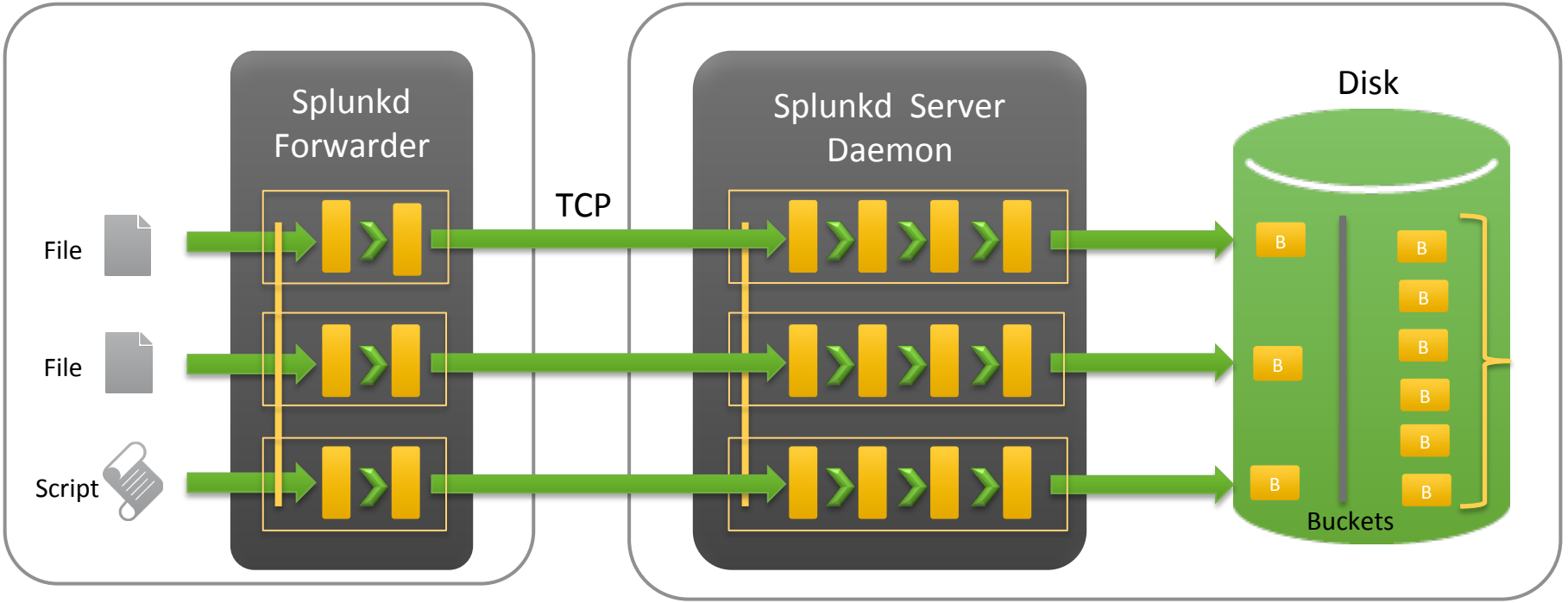
- $SPLUNK_HOME/etc/system/local/server.conf

```
[general]
parallelIngestionPipelines = 3
```

Multiple Ingestion Pipeline Sets – Details

- Each Pipeline Set has its own set of Queues, Pipelines and Processors
  - Exceptions are Input Pipelines which are usually singleton
- No state is shared across Pipeline Sets
- Data from a unique source is handled by only one Pipeline Set at a time

Multiple Ingestion Pipeline Sets over Network



Forwarder with 3 Pipeline Sets

Indexer with 3 Pipeline Sets

Multiple Ingestion Pipeline Sets – Monitor Input

- Each Pipelineset has its own set of TailReader, BatchReader and Archive Processor
- Enables parallel reading of files and archives on Forwarders
- Each file/archive is assigned to one pipeline set

Multiple Ingestion Pipeline Sets - Forwarding

- Forwarder:
  - One tcp output processor per pipeline set
  - Multiple tcp connections from the forwarder to different indexers at the same time
  - Load balancing rules applied to each pipeline set independently
- Indexer:
  - Every incoming tcp forwarder connection is bound to one pipeline set on the Indexer

Multiple Ingestion Pipeline Sets - Indexing

- Every pipeline set will independently write new data to indexes
- Data is written in parallel to better utilize resources
- Buckets produced by different pipeline sets could have overlapping time ranges

Search : Parallelization Efforts Performance Improvements

splunk>

Search Parallelization: Performance Improvement

Splunk Searches are faster in 6.3.

- Parallelizing the Search Pipeline
- Improving the Search Scheduler
- The Summary Building is parallelized and faster.

Search Pipeline

Batch Search

Reading Order

Reading Order

Option1:…B3 B5 B1 B2 B1 B6

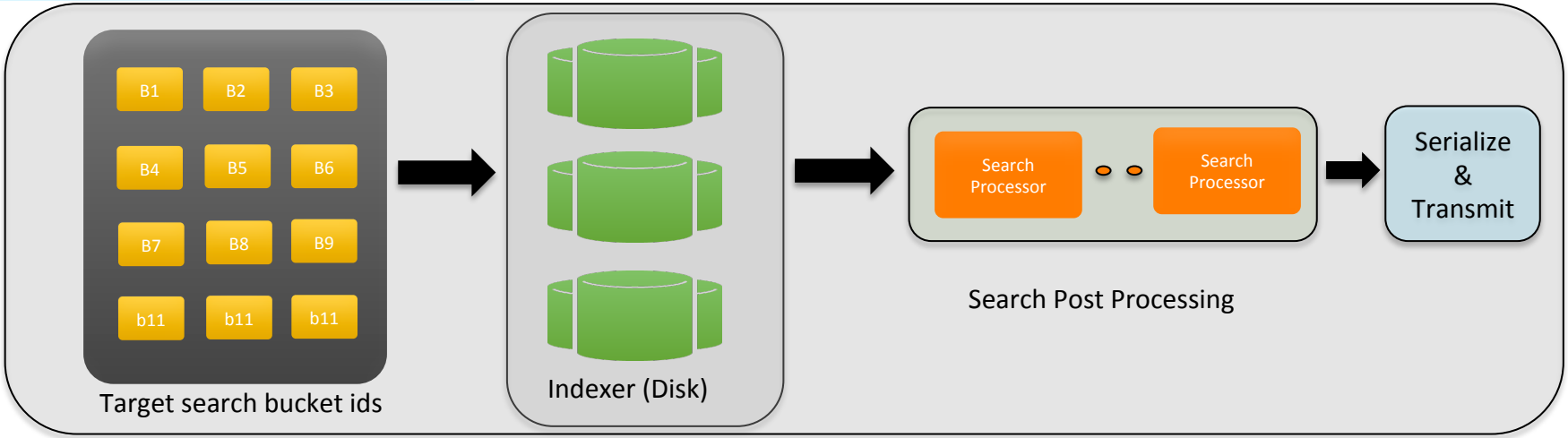
Option2:…B6 B5 B4 B3 B2 B1

Option 3…B6 B5 B4 B7 B4 B9

- Cursored Search: Time ordered data retrieval
- Batch Search: Bucket ordered data retrieval

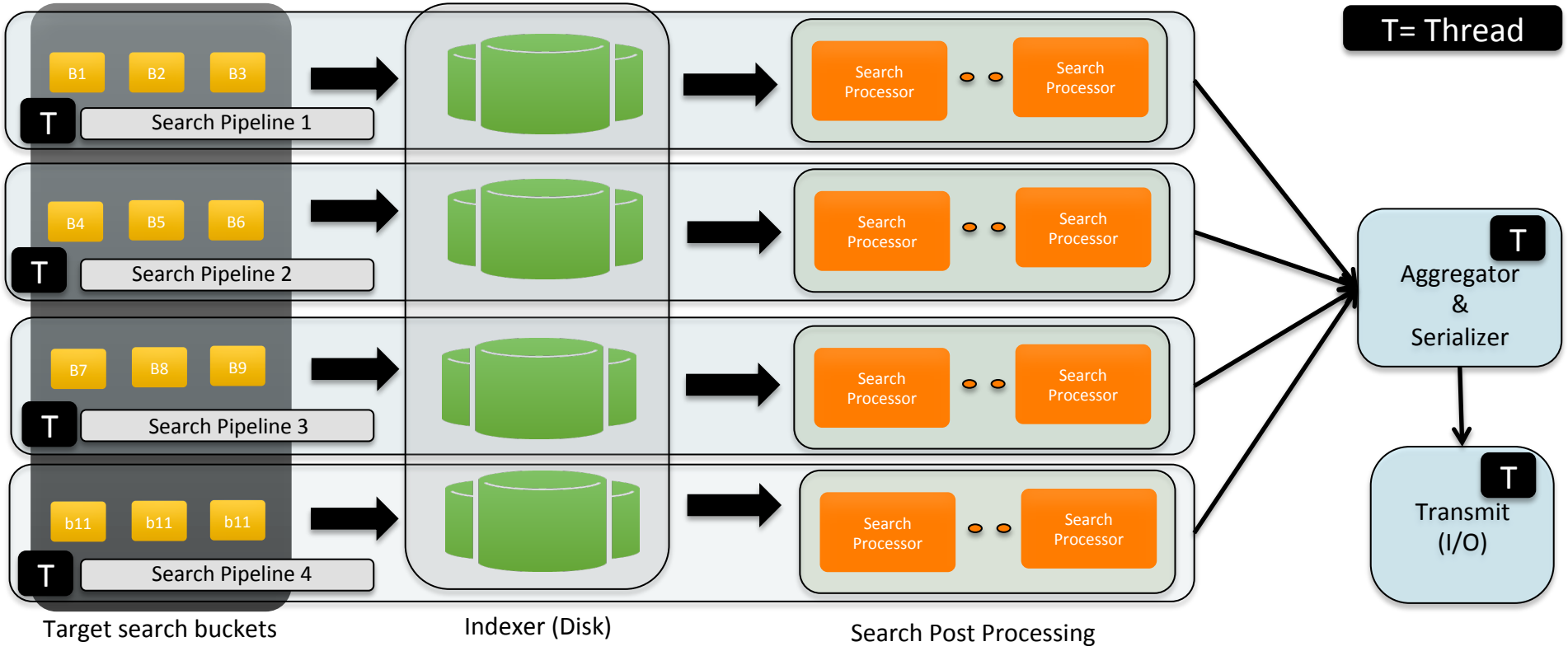
Iterates over buckets, creates a cursor and hence needs to time ordering is not tracked

Facilitates parallel processing of buckets independently across multiple pipeline



Search Pipeline at the Peer

Batch Search: Pipeline Parallelization



Batch Search: Pipeline Parallelization

- Under-utilized indexers provide us opportunity to execute multiple search pipelines
- Batch Search time-unordered data access mode is ideal for multiple search pipelines
- No state is shared i.e. no dependency exists across Search Pipelines.
- Peer/Indexer side optimizations
- Takeaway :
  - Under utilized indexers are candidates for search pipeline parallelization.
  - Do NOT enable if indexers are loaded

Configuring the Batch Search in Parallel mode

- How to enable?

  $SPLUNK_HOME/etc/system/local/limits.conf

```
[search]
batch_search_max_pipeline  = 2
```

- What to expect?

  Search performance in terms of retrieving search results improved.

  Increase in number of threads

Search Scheduler Improvements

- Scheduler improvements in Splunk Enterprise 6.3:
  - Priority Scoring
  - Schedule Windows
- Performance improvements over previous schedulers
  - Lower Lag
  - Fewer skipped searches

Search Scheduler Improvements
Priority Score

Problem in 6.2:

Simple single-term priority scoring could result in saved search lag, skipping, and starvation (under CPU constraint).

Solution in 6.3:

Better multi-term priority scoring mitigates problems and improves performance by 25%.

score(j) = next_runtime(j)
+ average_runtime(j) × priority_runtime_factor
– skipped_count(j) × period(j) × priority_skipped_factor
+ schedule_window_adjustment(j)

Search Scheduler Improvements

Problem in 6.2

Scheduler can not distinguish between searches that (A) *really should* run at a specific time (just like cron) from those that (B) don't have to. This can cause lag or skipping.

Solution in 6.3:

Give a *schedule window* to searches that don't have to run at specific times.

Example:

For a given search, it's OK if it starts running sometime between midnight and 6am, but you don't really care when specifically.

- A search with a window helps *other* searches.
- Search windows *should not* be used for searches that run every minute.
- Search windows *must* be less than a search's period

Configuring Search Scheduler

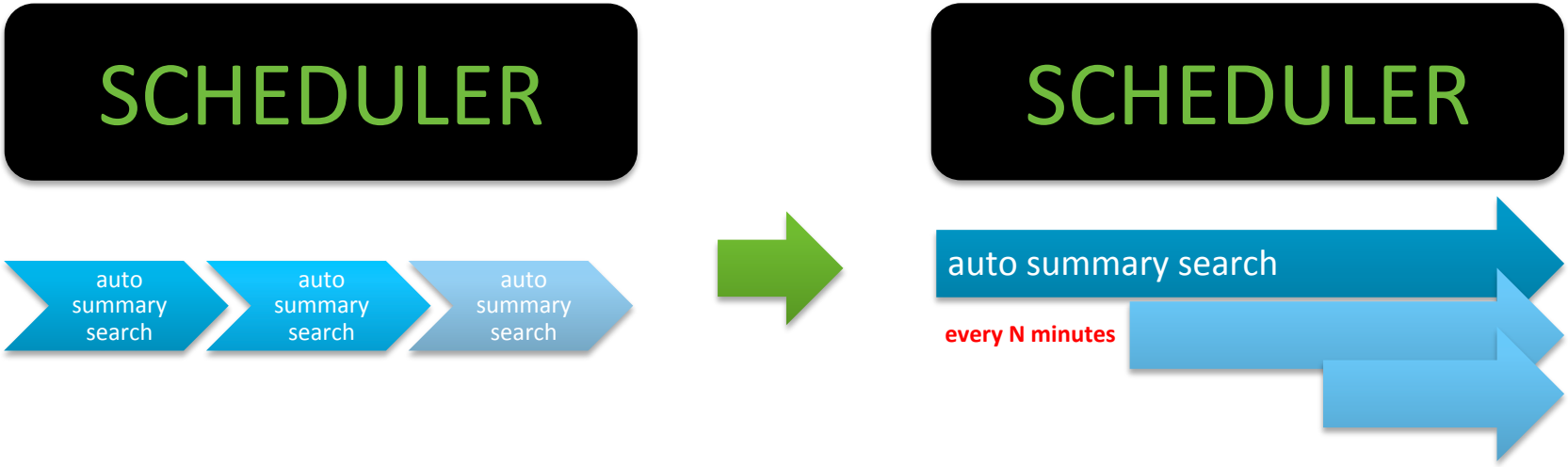
$SPLUNK_HOME/etc/system/local/limits.conf

```
[scheduler]
max_searches_perc = 50

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

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

Search: Parallel Summarization

- Sequential nature of building summary data for data model and saved reports is slow
- Summary Building process has been parallelized in 6.3

Summary Building Parallelization



Sequential Summary Building

Parallelized Summary Building

Configuring Summary Building for Parallelization

- $SPLUNK_HOME/etc/system/local/savedsearches.conf

```
[default]
auto_summarize.max_concurrent = 2
```

- $SPLUNK_HOME/etc/system/local/datamodels.conf

```
[default]
acceleration.max_concurrent = 2
```

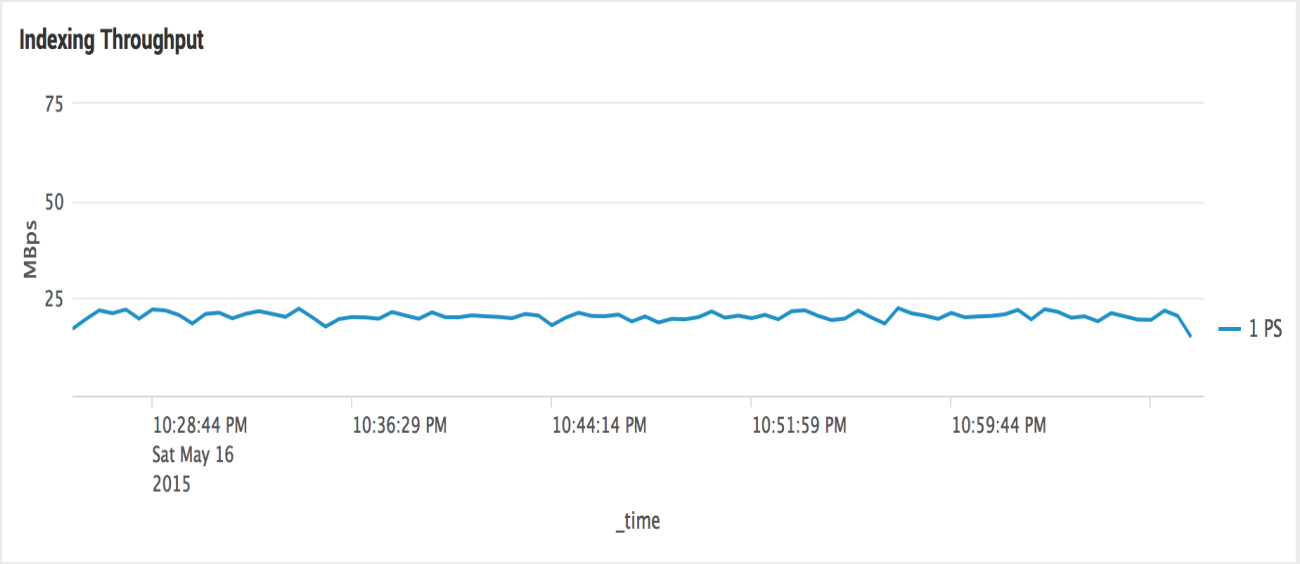
6.3 Performance

splunk>

6.2 vs. 6.3 Performance Tests

- System Info
  - Dell PowerEdge R820
  - Intel(R) Xeon CPU E5-4620 @ 2.20 GHz
  - 32 cores w/o Hyper-Threading
  - 128 GB RAM
  - 8 x 146GB SAS 15k RPM disks in RAID-10
  - 1 Gb Ethernet NIC
  - CentOS 6
- No other load on the box

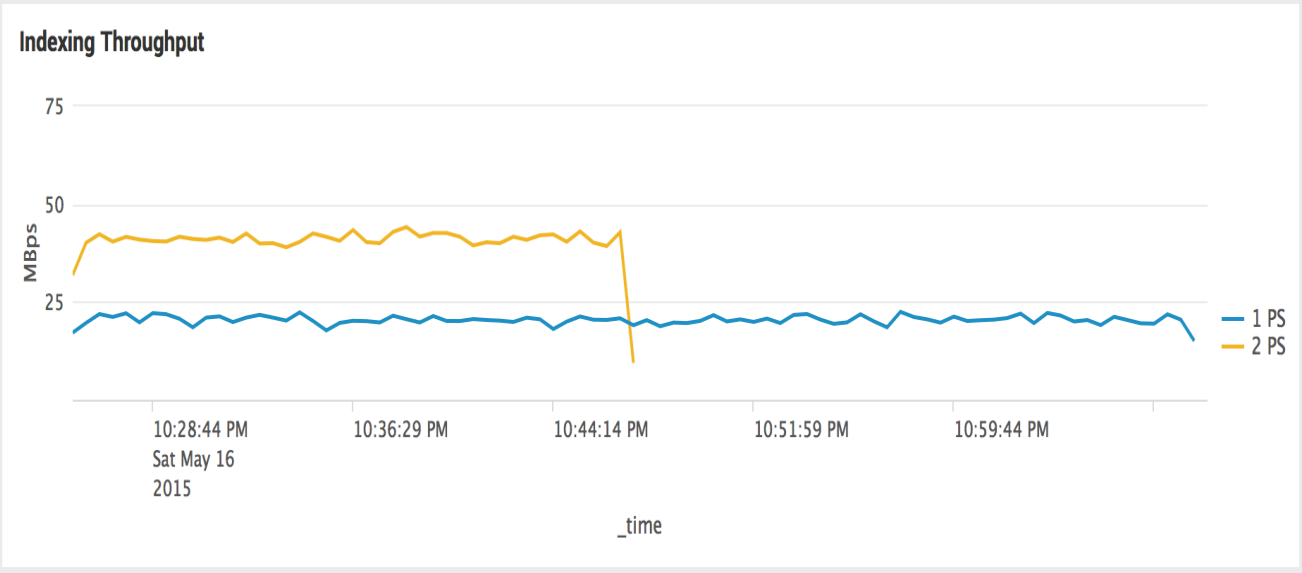
Indexing Tests



Pipelines	Average Throughput (MB/s)
1	20.02
2	
3	
4	

- Index a 50 GB generic syslog dataset. No search loads.
- Time to Index – 41.23 minutes
- Single Pipeline on 6.2 or 6.3 will have similar performance

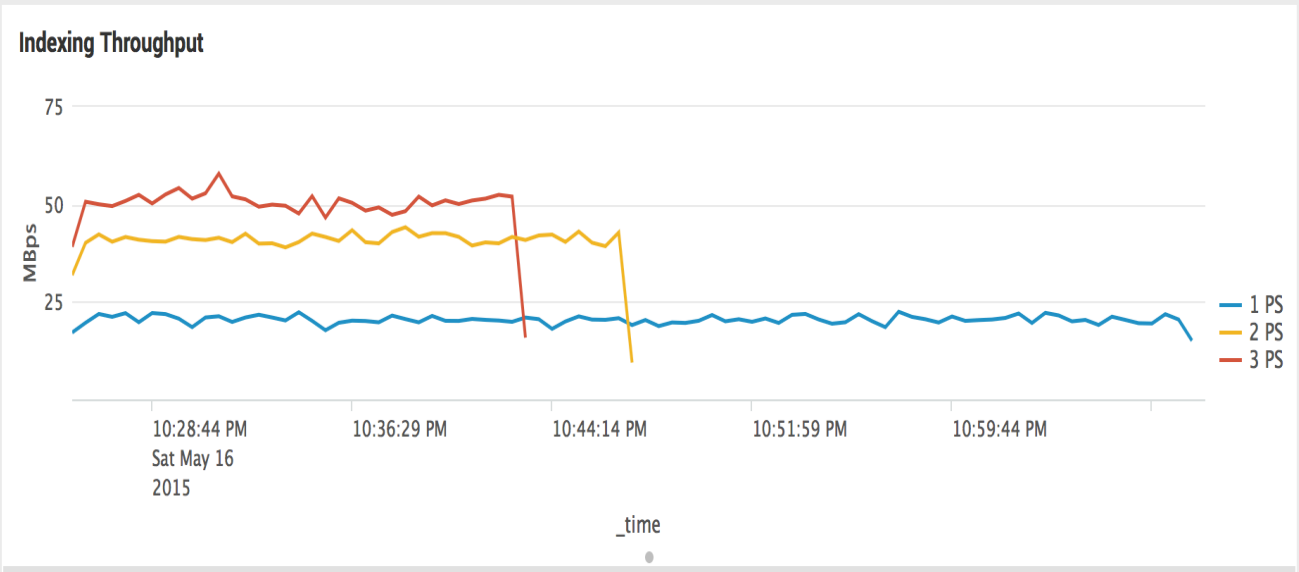
Indexing Tests



Pipelines	Average Throughput (MB/s)
1	20.02
2	40.04
3	
4	

- Time to Index 50 GB – 20.81 minutes
- 98.2 % Increase in Average Indexing Throughput
- On an average Splunk utilized 2x CPU cores , 1.3x Memory and 2x Disk IOPS

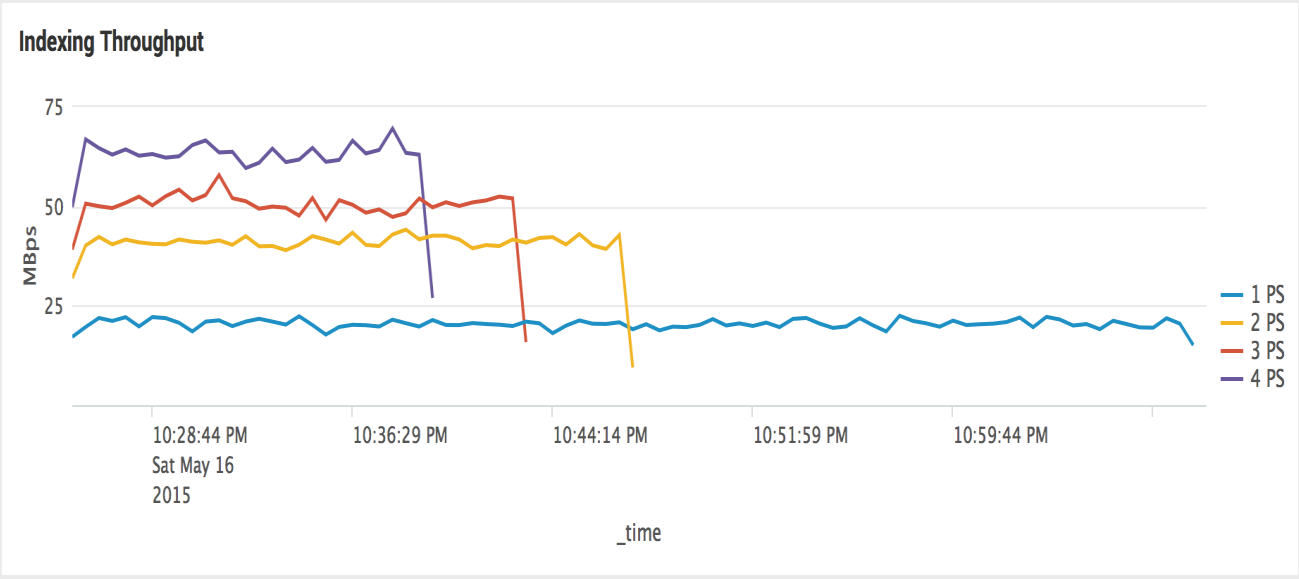
Indexing Tests



Pipelines	Average Throughput (MB/s)
1	20.02
2	40.04
3	50.91
4	

- Time to Index 50 GB – 16.63 minutes
- 152 % Increase in Average Indexing Throughput
- On an average Splunk utilized 3x CPU cores, 1.6x Memory and 2.5x Disk IOPS

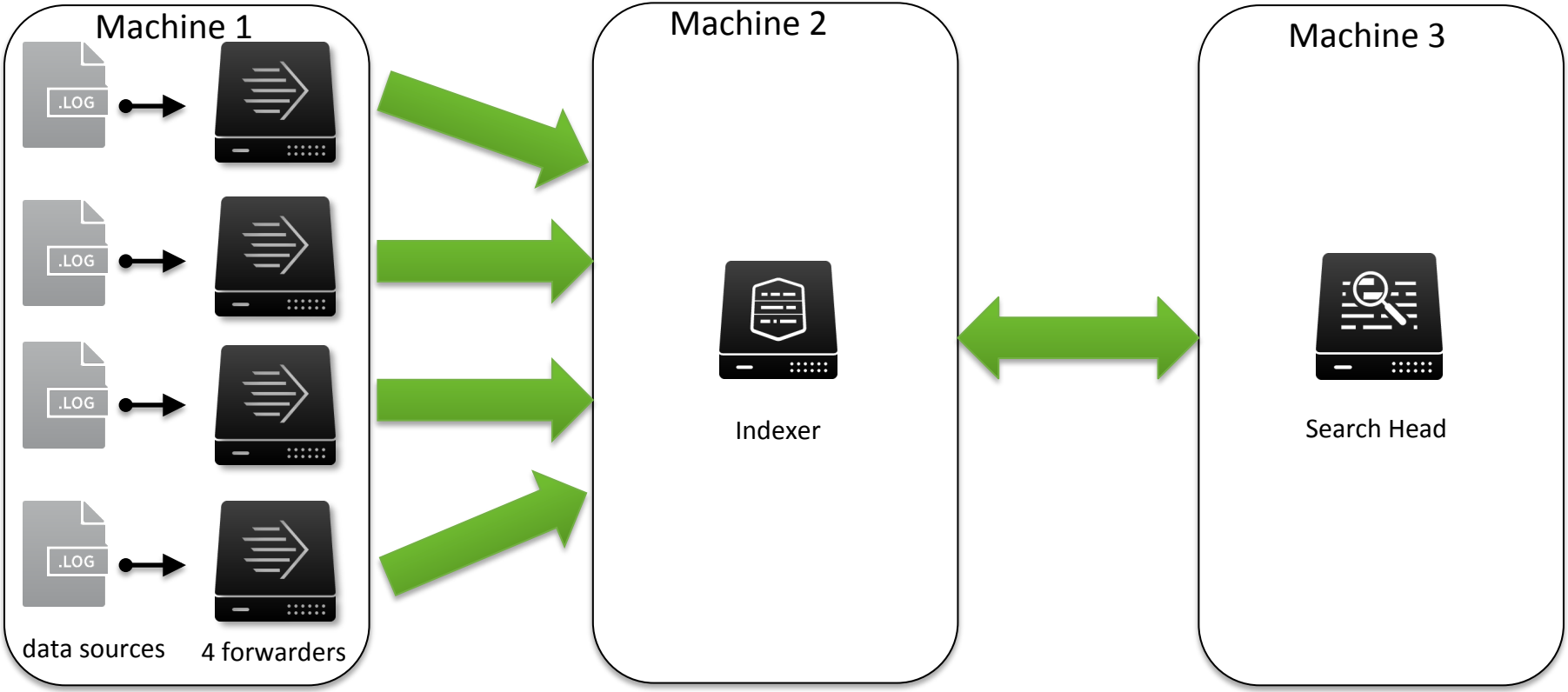
Indexing Tests



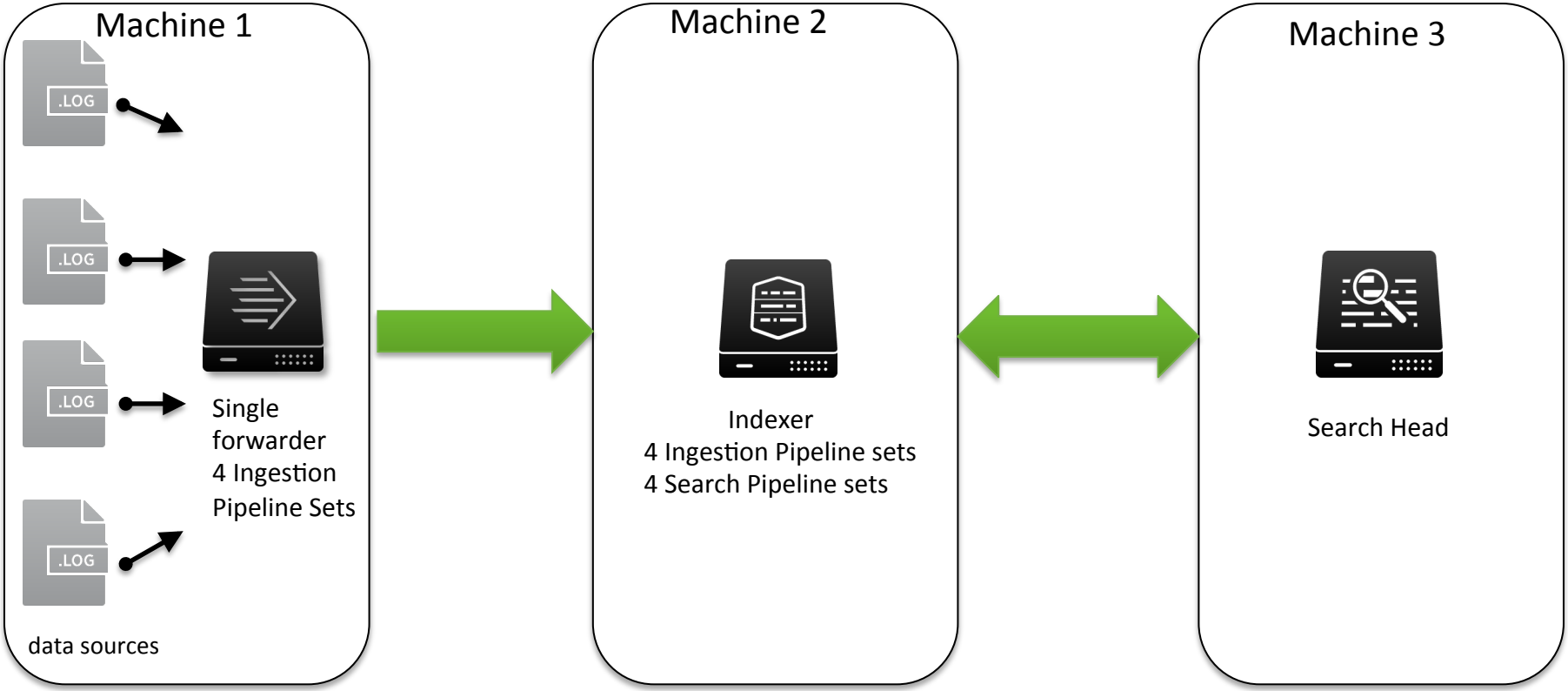
Pipelines	Average Throughput (MB/s)
1	20.02
2	40.04
3	50.91
4	62.07

- Time to Index 50 GB – 13.42 minutes
- 207% Increase in Average Indexing Throughput
- On an average Splunk utilized 4x CPU cores, 2.25x Memory and 3x Disk IOPS

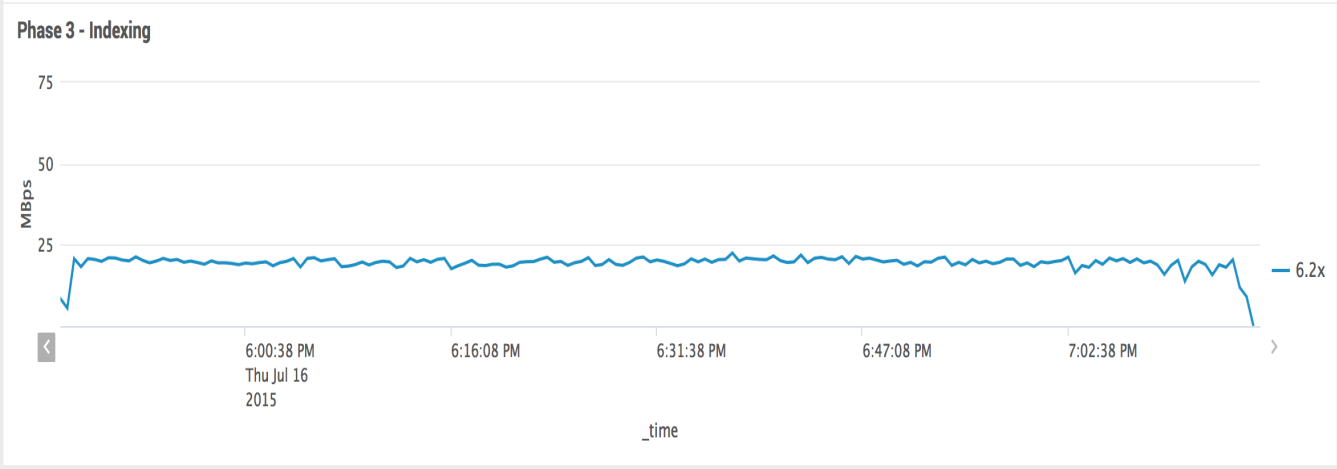
6.2 Setup


Machine 1
.LOG
.LOG
.LOG
.LOG
data sources
4 forwarders
Machine 2
Indexer
Machine 3
Search Head

6.3 Setup


Machine 1
.LOG
.LOG
.LOG
.LOG
Single forwarder
4 Ingestion Pipeline Sets
data sources
Machine 2
Indexer
4 Ingestion Pipeline sets
4 Search Pipeline sets
Machine 3
Search Head

Burst in Indexing Load + Searches

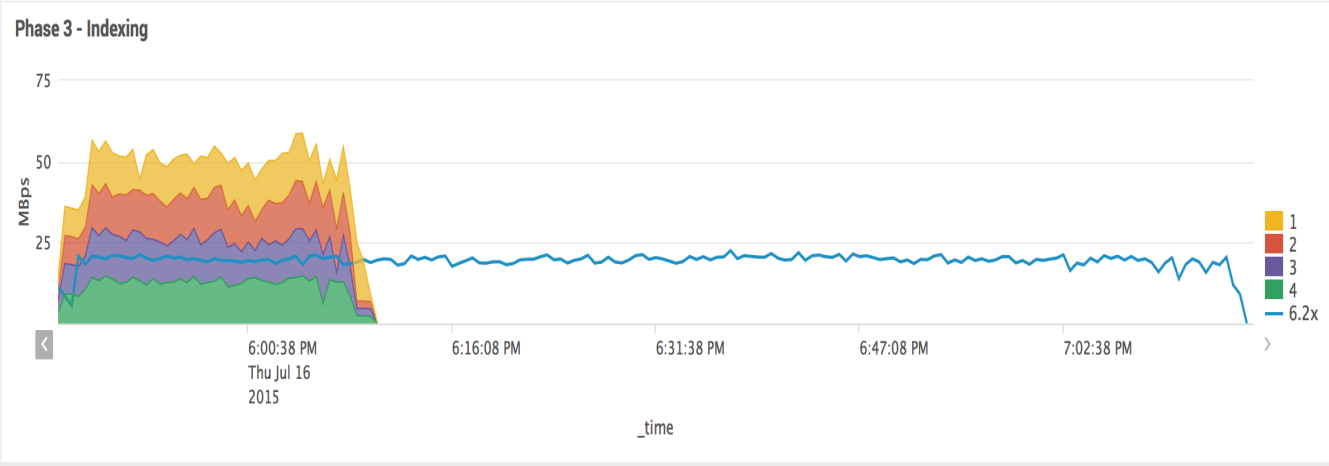


Version	Average Throughput (MB/s)
6.2	20.0
6.3	

Splunk 6.2

- Data forwarded @ 9 MB/s + Monitor 50GB dataset
- 89.8 Minutes to Index this 50 GB dataset
- Number of Ingestion Pipelines - 1
- Number of Concurrent Searches – 4

Burst in Indexing Load + Searches



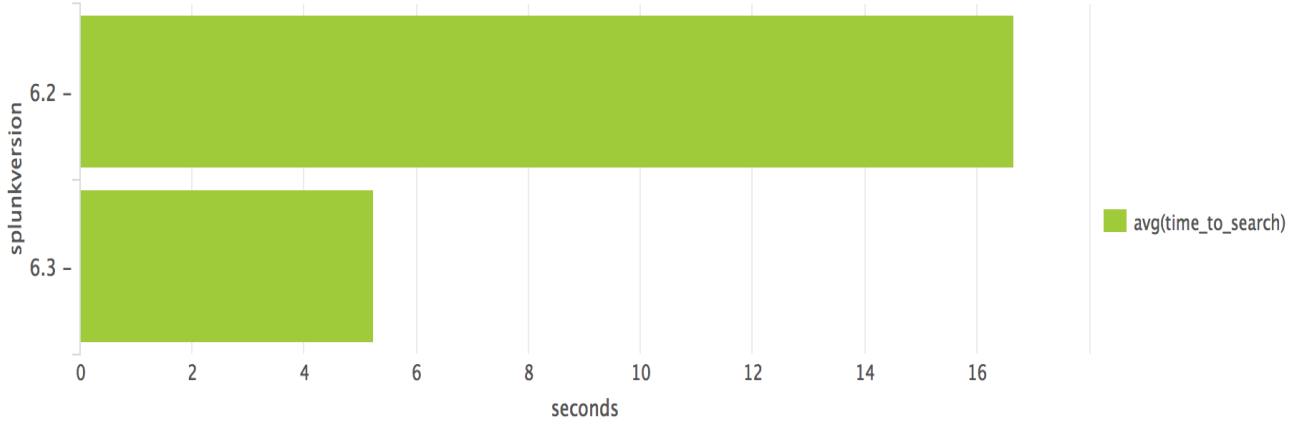
Version	Average Throughput (MB/s)
6.2	20.0
6.3	48.5

Splunk 6.3

- Data forwarded @ 9 MB/s + Monitor 50GB dataset
- 23.4 Minutes to Index this 50 GB dataset
- 142% Increase in Average Indexing Throughout
- Number of Ingestion Pipelines - 4
- Number of Concurrent Searches – 4

Batch Mode Sparse Search

index=test_static every1k | stats count

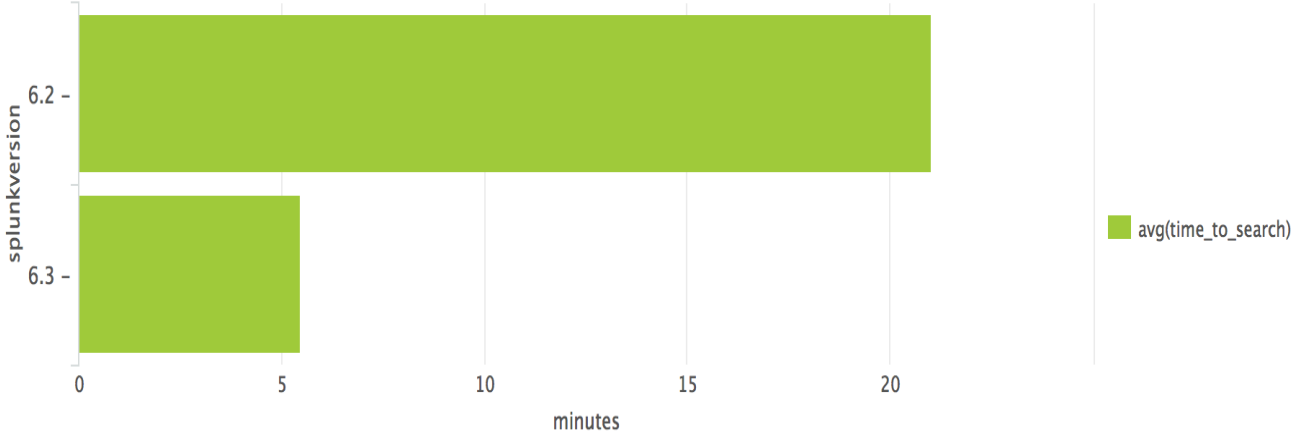


Version	Average Search Time (seconds)
6.2	16.67 s
6.3	5.24 s

- Sparse Search – Characterized predominately by returning some events per bucket
- Splunk 6.2 – 1 Search Pipelines
- Splunk 6.3 – 4 Search Pipelines
- Search is 3.18x faster in 6.3

Batch Mode Dense Search

index=test_static every1 | stats count



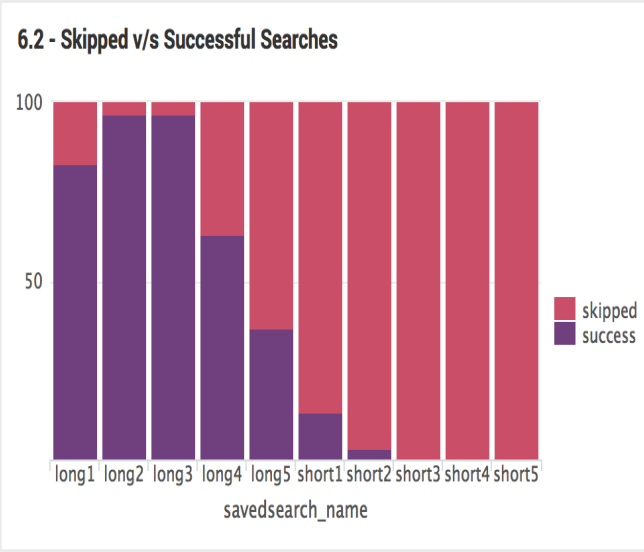
Version	Average Search Time (minutes)
6.2	21.0 m
6.3	5.46 m

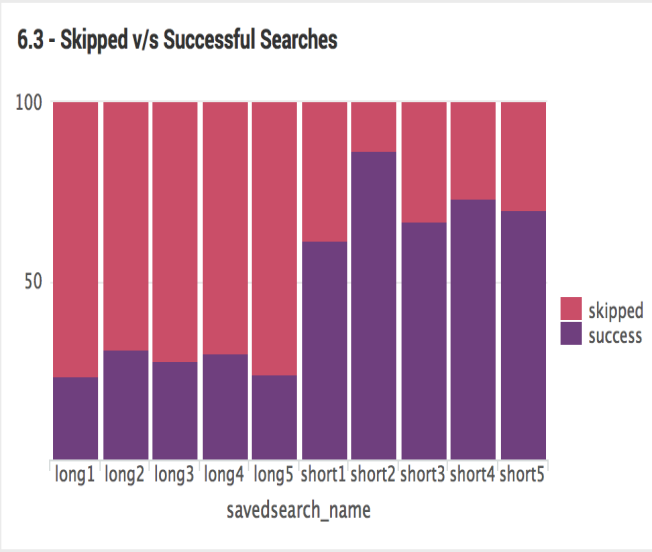
- Dense Search – Characterized predominately by returning many events per bucket
- Splunk 6.2 – 1 Search Pipelines
- Splunk 6.3 – 4 Search Pipelines
- Search is 3.85x faster in 6.3

Scheduled Searches Setup

- 10 Searches are scheduled to run every minute
- 5 Longer running searches (~40s)
- 5 Shorter running searches (~15s)
- Test configured to run only 3 scheduled concurrently

Scheduled Searches

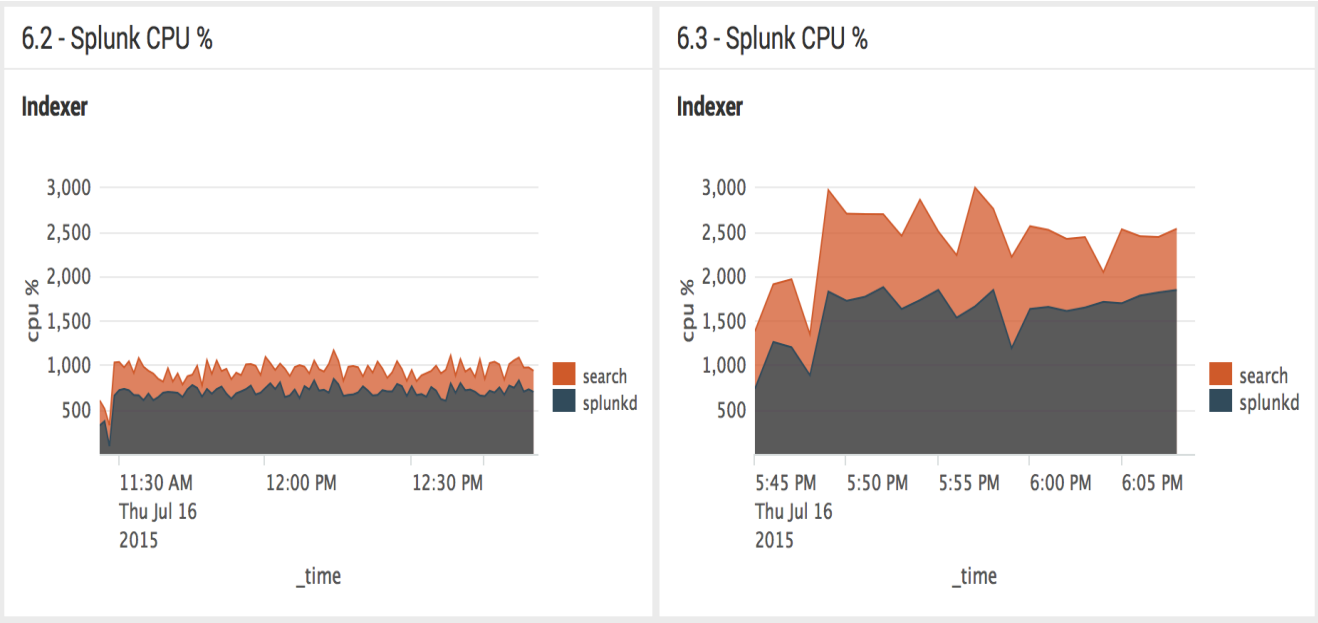




Version	Searches completed
6.2	118
6.3	148

- Skipped vs. Successful Searches – 30 minute window
- 25% Increase in Successful Searches
- This optimization will not utilize additional System Resource

CPU Utilization



Average CPU %

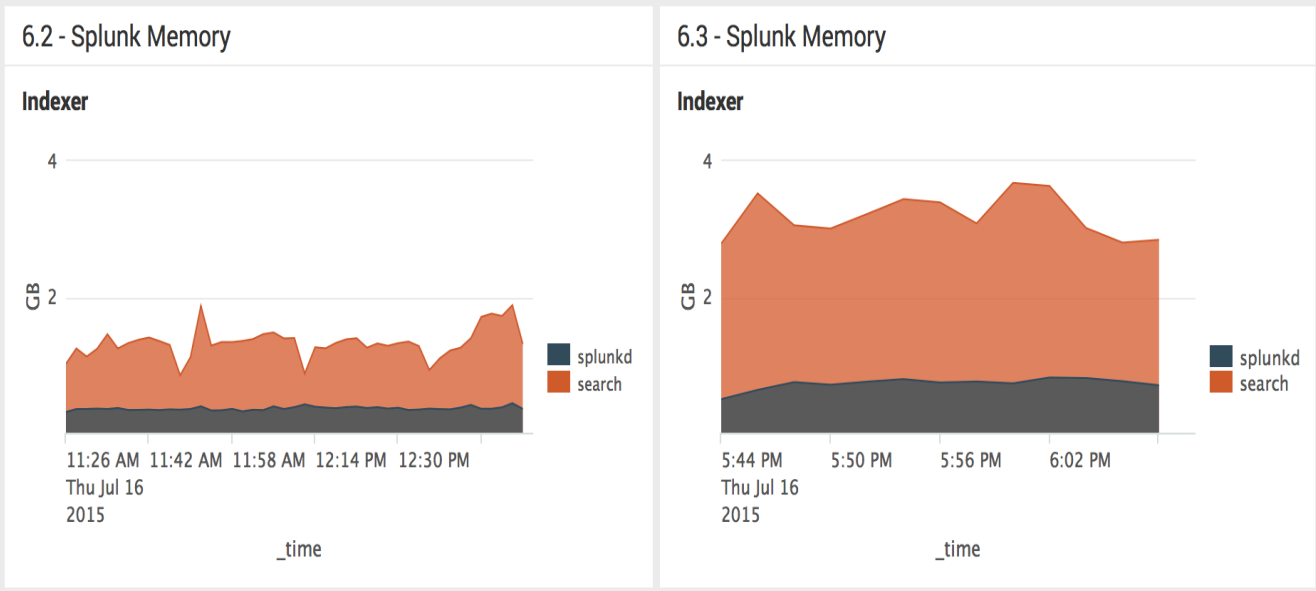
6.2	6.3
882 %	2237 %

Burst in Indexing Load + Searches

Splunk 6.2 – 1 Ingestion Pipeline ; 1 Search Pipeline

Splunk 6.3 – 4 Ingestion Pipelines ; 4 Search Pipelines

Memory Utilization



Average Memory

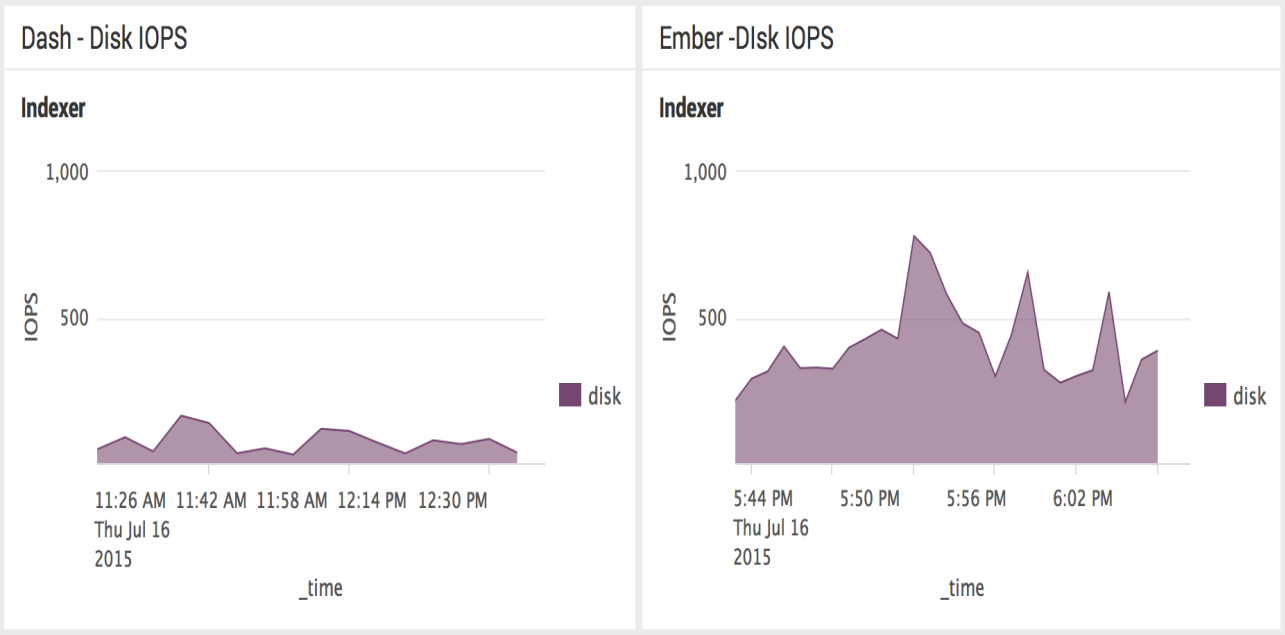
6.2	6.3
1.18 GB	2.69 GB

Burst in Indexing Load + Searches

Splunk 6.2 – 1 Ingestion Pipeline ; 1 Search Pipeline

Splunk 6.3 – 4 Ingestion Pipelines ; 4 Search Pipelines

Disk Utilization



Average Disk IOPS

6.2	6.3
157	483

Burst in Indexing Load + Searches

Splunk 6.2 – 1 Ingestion Pipeline ; 1 Search Pipeline

Splunk 6.3 – 4 Ingestion Pipelines ; 4 Search Pipelines

Final Thoughts

- What is my current workload?
  - Data volume – Daily and Peak
  - Search Volume – Concurrent and total
  - System Resource Usage
- How do I approach these features?
  - System significantly under-utilized ?
  - Search Pipelines
    - Lot of Batch mode Searches ?
  - Parallel Ingestion Pipelines
    - Handling Bursts in Data?
    - Reading large number of files in parallel?
- Splunk scales horizontally

THANK YOU

splunk>