Splunk SmartStore

NextGen Data Architecture with a customer perspective

Bharath Aleti — Product Management, Splunk
Jeff Champagne — Field Architect, Splunk
Jeremiah Cutting — Senior Staff IT Engineer, Qualcomm

October 2018

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

Agenda

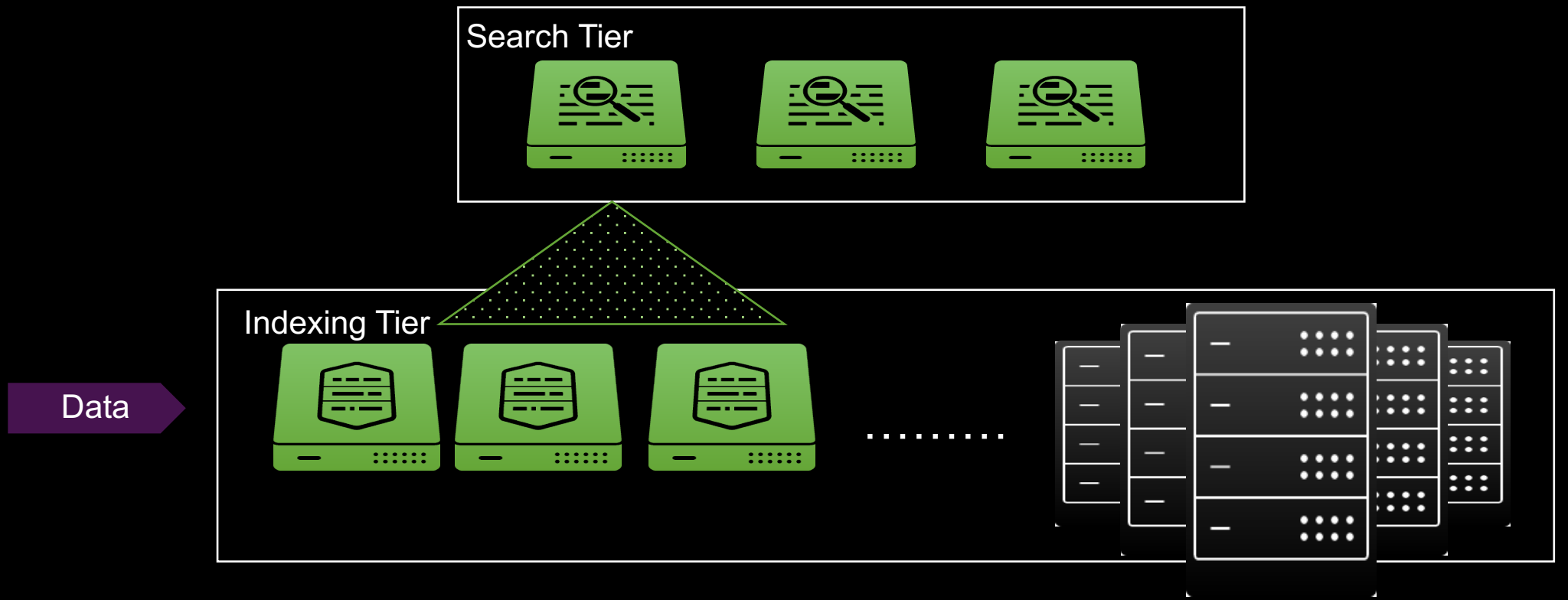
- Introducing SmartStore ?
- How does SmartStore work ?
- Customer Journey

Introducing SmartStore

Next Generation Data Architecture

splunk> .conf18

Growing data volumes requires $$$ infra spend



Adding new indexers in response to data growth is expensive => High cost
Searches typically run over only on a partial subset of data => Inefficient utilization

Alternative

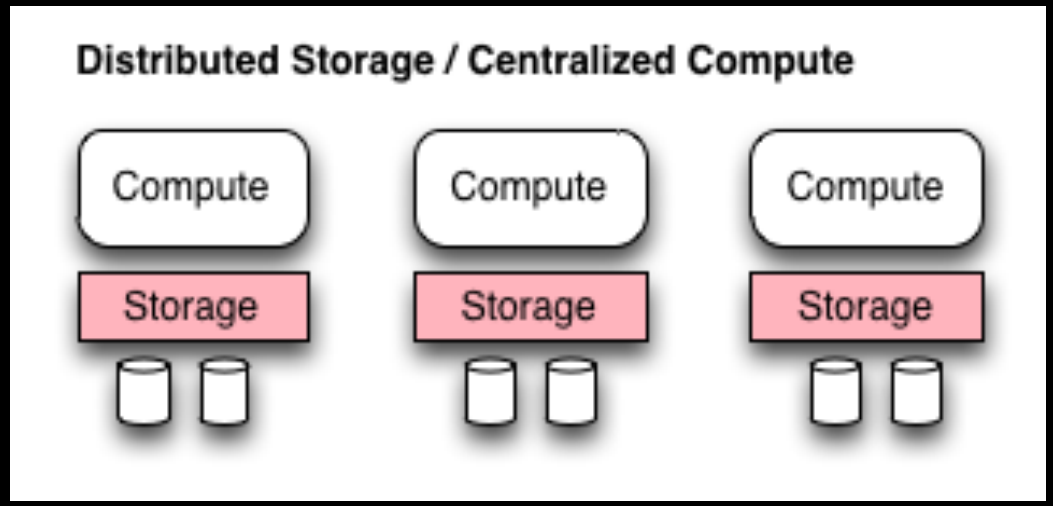
What IF

- Splunk provided a low-cost way to store data
- AND provided elasticity and flexible capacity
- AND ensured performance for diverse workloads
- AND allowed rapid scaling and deployment

How does this help customers?

- Analyze growing volumes of data, at a significantly lower cost

Introducing SmartStore

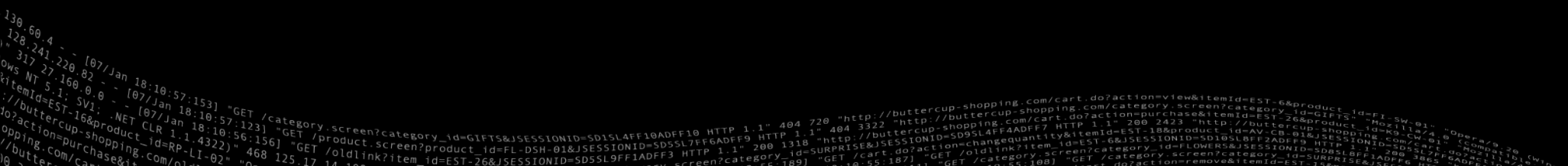


- SmartStore decouples your storage and compute
- Elastically scale compute on-demand
- Independently grow storage to accommodate retention
- Achieve cost savings with more flexible compute/storage options

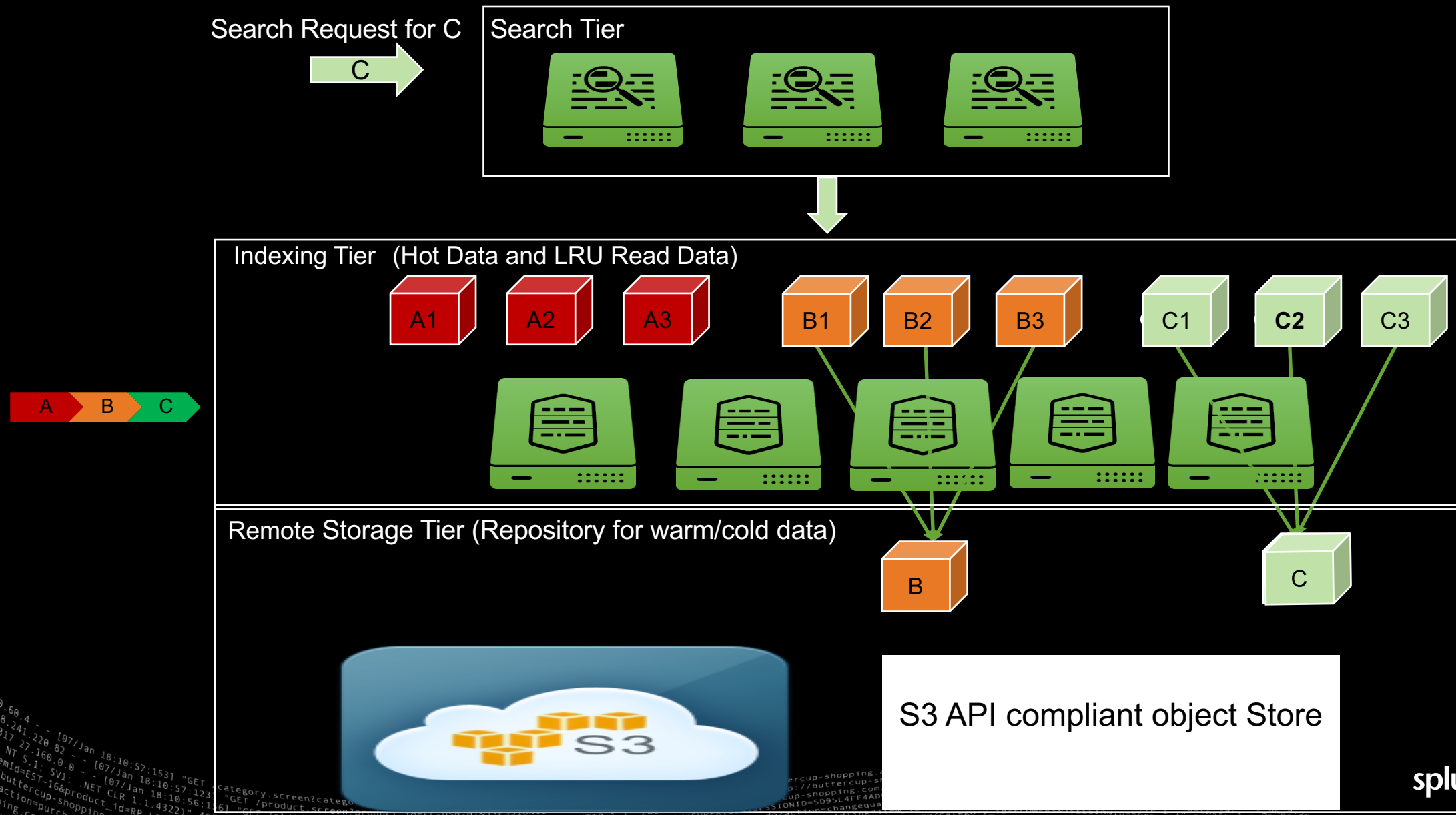
Lower TCO by providing ability to scale storage and compute independently

SmartStore Architecture

- Local Storage
  - Hot buckets are always on local storage [homePath]
    - No change from classic architecture
- Remote Storage
  - Buckets are copied to the [remotePath] when they roll from Hot → Warm
    - Remote storage must provide data protection
    - Splunk does not provide resiliency for buckets in remote storage
- Cache Manager
  - Recently read buckets are also on local cache
    - New indexer functionality (not a new role)
    - Each indexer has a cache manager that operates independently
    - Retrieves buckets from the remote store when needed
    - Evicts buckets from the local cache [homePath]



SmartStore Overview



SmartStore Benefits

Scalability & High Availability

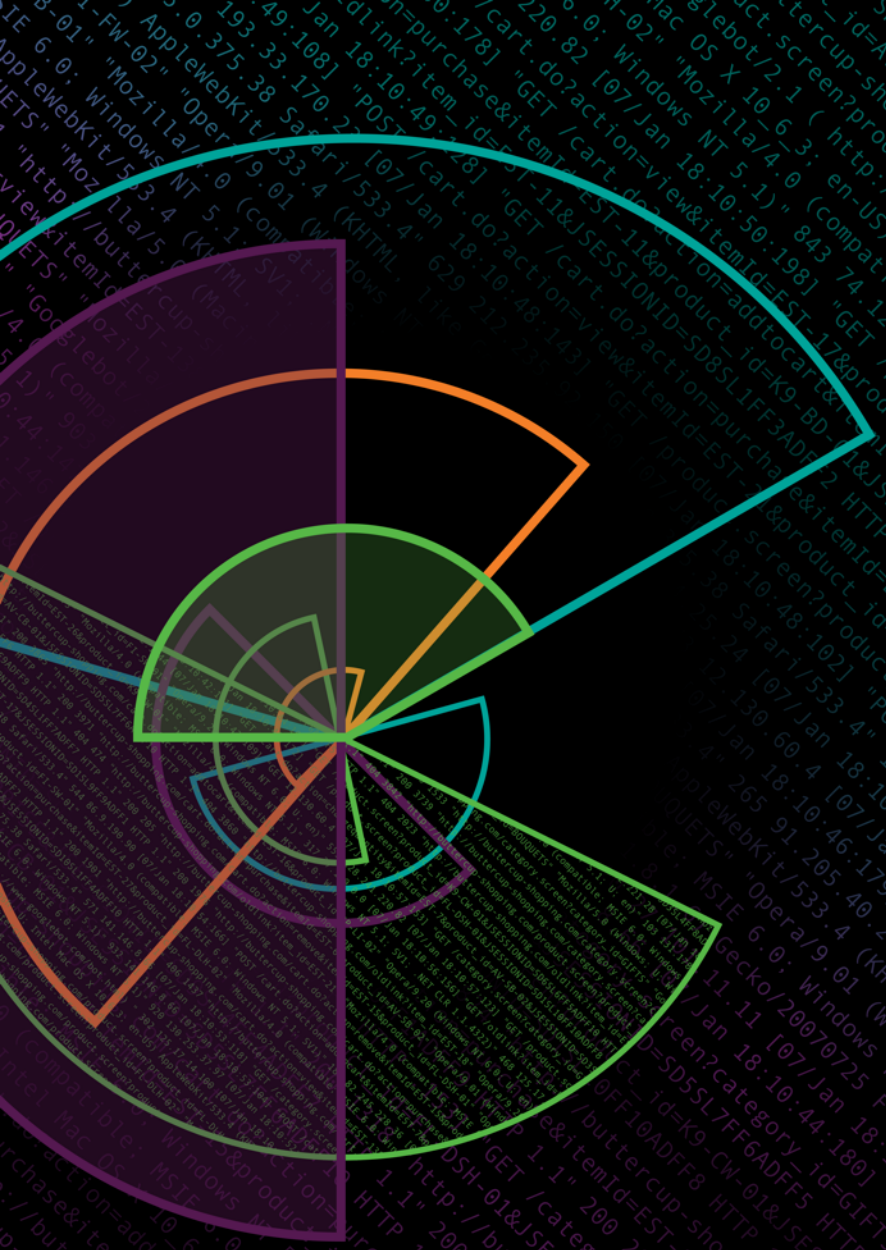
- Architectured for massive scale
- High data availability with remote storage tier
- Performance at scale with cached active dataset

TCO Reduction

- Scale compute and storage independently
- Lower TCO with reduced indexer footprint
- Leverage cost benefits of cloud/storage innovations

Simplified Management

- Instant indexer failure recovery
- Faster data rebalance
- Upgrade/replace indexer infrastructure with simple bootstrap from remote store
- New global size based retention policies

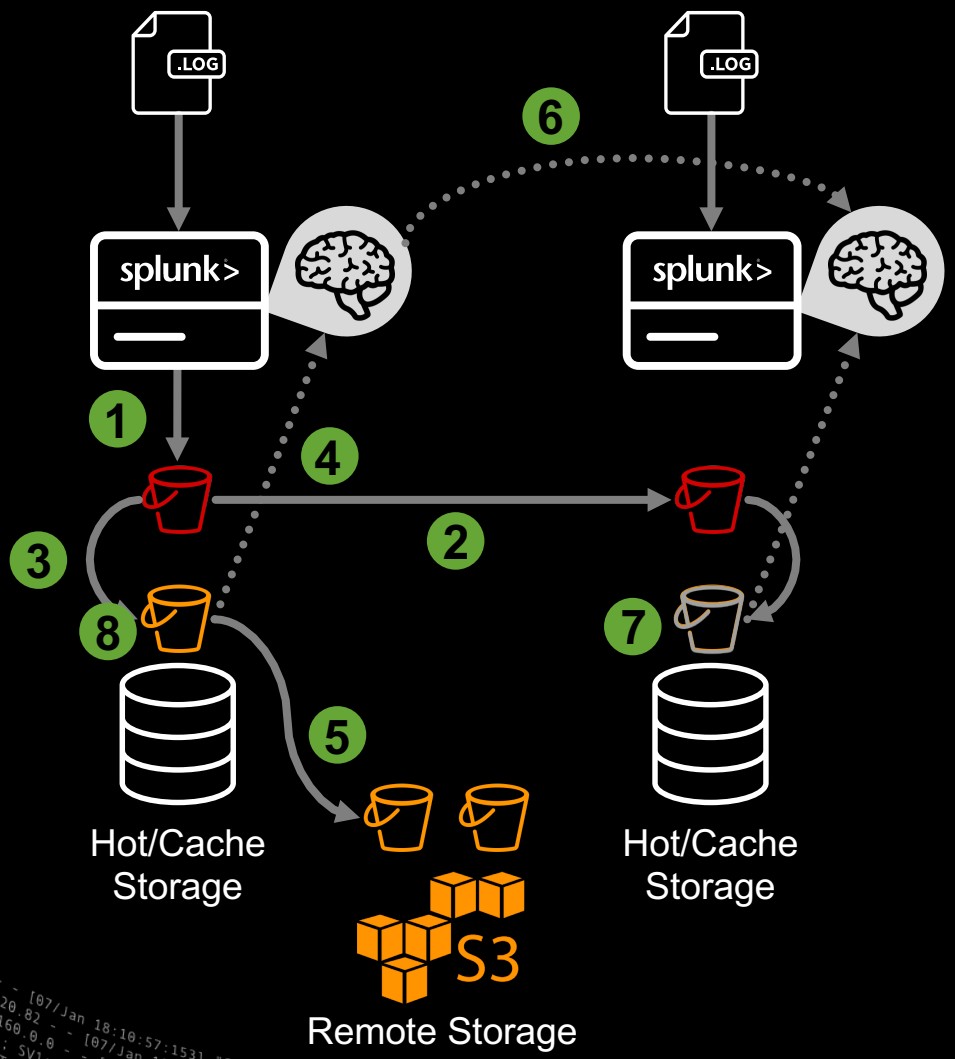


How does SmartStore work ?

How does data ingestion and search work ?

Getting Data In

Clustered Deployments



1. Data arrives and is written to a Hot bucket
2. Hot bucket streams to cluster peer(s) according to RF
3. Replication completes and the buckets roll to warm
4. Buckets are registered with their cache managers
5. Cache manager on source peer uploads the bucket to the remote store
6. Source peer notifies replication peers that the bucket was uploaded successfully
7. Replication peers delete their local copy of the bucket and retain a stub
8. Cached copy remains on the source peer until evicted by the local cache manager

Searching with SmartStore

Search Facts

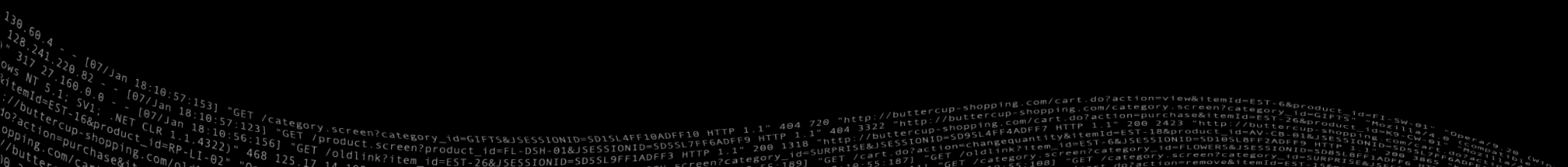
Some assertions about Splunk searches…

- Typically over near-term data
  - Research has shown that 97% of searches look back 24hrs or less

By default, the cache manager will attempt to cache buckets with recent events

- Typically have spatial and temporal locality
  - *If I find an event at a specific time or in a log, I will likely run additional searches against data at that time or in that log*

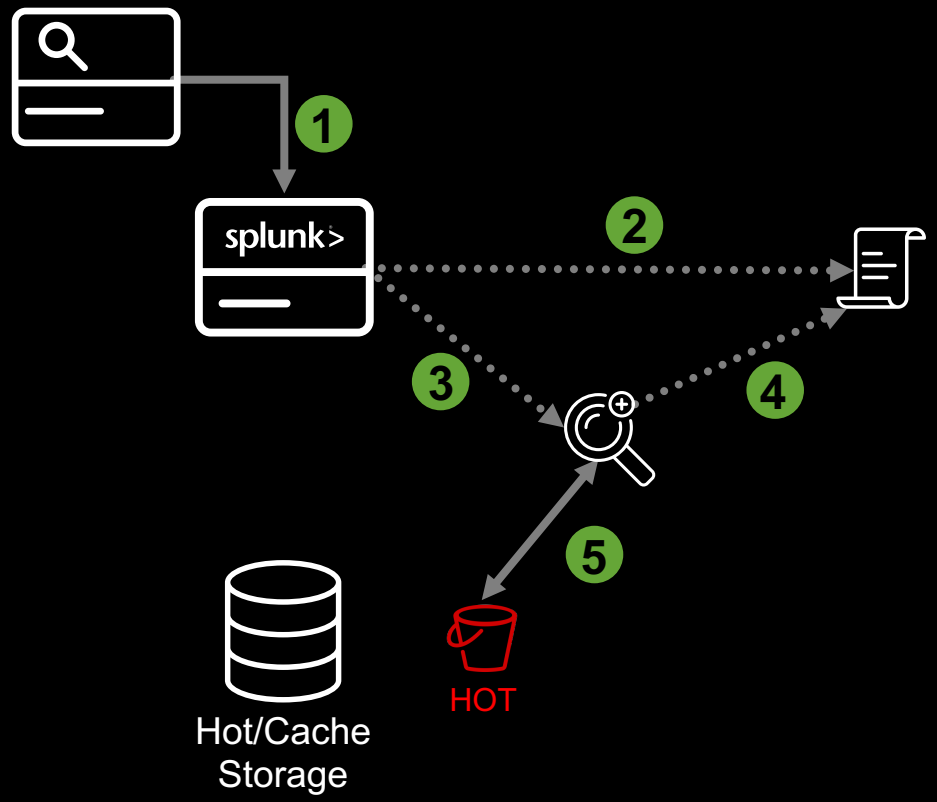
By default, the cache manager will attempt to cache recently accessed buckets



splunk> .conf18

Searching with SmartStore

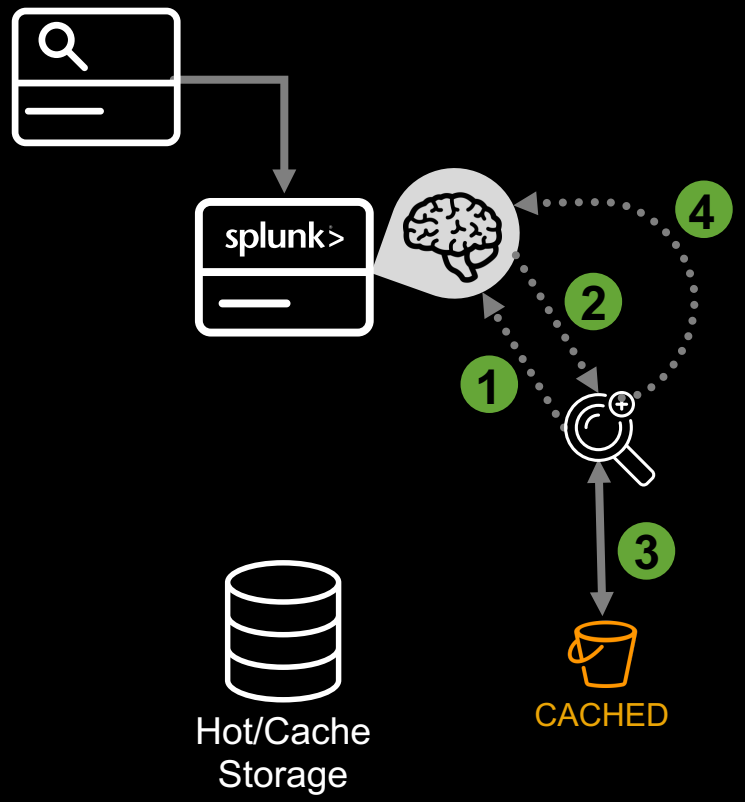
Hot Buckets



1. Search request is received
2. Indexer generates a list of relevant buckets to be searched
3. Search process is spawned
4. Spawned process reads the bucket list
5. Hot buckets are searched in the same manner as “classic” search

Searching with SmartStore

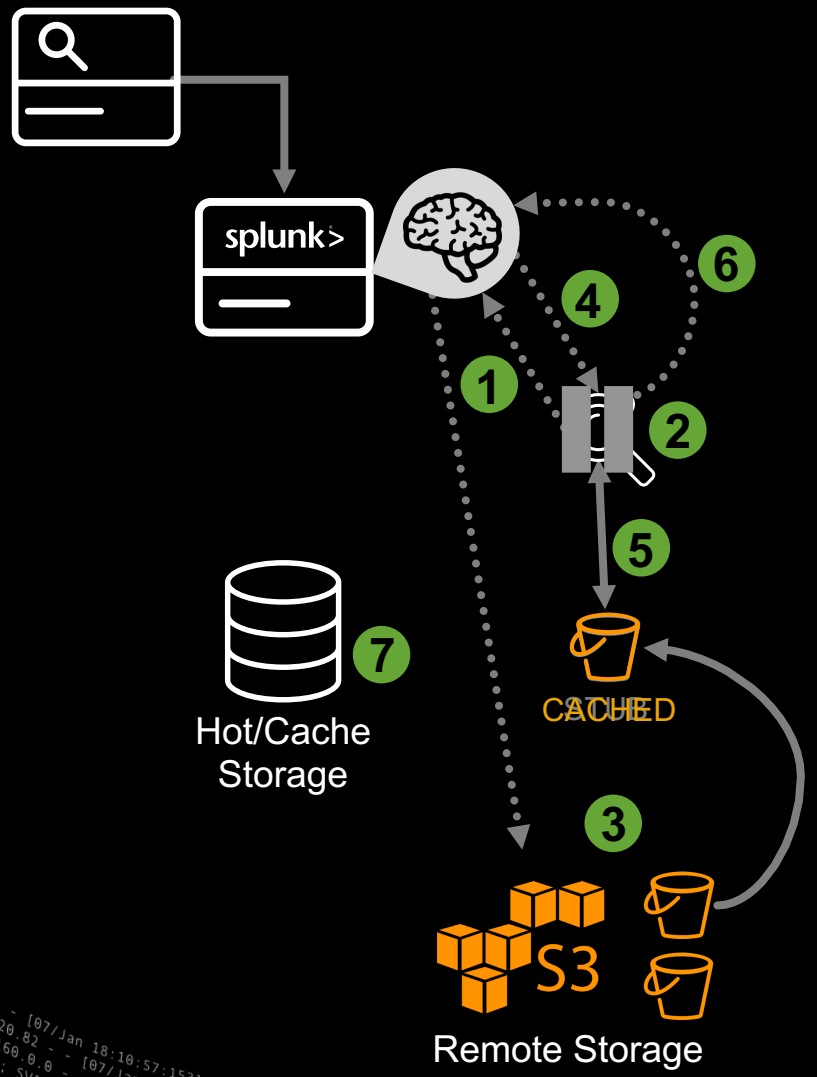
Cached Buckets



1. Search process “opens” the bucket with the Cache manager
2. Cache manager tells the search process that the bucket is local and available for search
3. Search process searches the bucket
4. Search process ”closes” the bucket with the cache manager

Searching with SmartStore

Remote Buckets



1. Search process “opens” the bucket with the Cache manager, but it isn’t in cache
2. Search process waits
3. Cache manager fetches the bucket from the remote store
4. Cache manager tells the search process that the bucket is local and available for search
5. Search process searches the bucket
6. Search process ”closes” the bucket with the cache manager
7. Bucket remains in cache until evicted by the cache manager

Cache Manager

Cached Data

When do we evict?

- Hot buckets are always local
- Warm buckets are not immediately evicted from the cache
  - Cache manager will attempt to keep buckets that contain events with timestamps from the last 24 hours [hotlist_recency_secs]
- TSIDX and Journals are evicted quicker than other bucket files
  - Cache manager will attempt to keep smaller bucket files for 15 days [hotlist_bloom_filter_recency_hours]

▶ Eviction Policies

Policy Name	Description
clock (default)	Prefer to evict bucket with the oldest events first, unless it has been accessed recently
lru	Evict the least recently used bucket
random	Randomly evict a bucket
lrult	Evict the bucket with the oldest events first
noevict	Don't evict – This can be used to provide data resiliency instead of indexer clustering

Cache Manager

Localizing Data

Cache manager offers the ability to fetch specific bucket files

- Ex: bloomfilter, TSIDX, metadata, journal

Some search commands only need specific files from the bucket

- Don't need the raw data
  - Ex: metadata, tstats
- Don't need any bucket content
  - Ex: eventcount, dbinspect

▶ Lookahead

- Cache manager will attempt to pre-fetch buckets needed for a search
  - Heuristic will adjust itself based upon the throughput from the remote store

High Availability

Data vs. Search

- Data Resiliency

Will my data be protected if something goes wrong?

  - SmartStore relies on the remote store to provide data resiliency
    - SmartStore and indexer clustering will not protect data in the remote store
    - When used on SmartStore enabled indexes, indexer clustering will only protect hot buckets

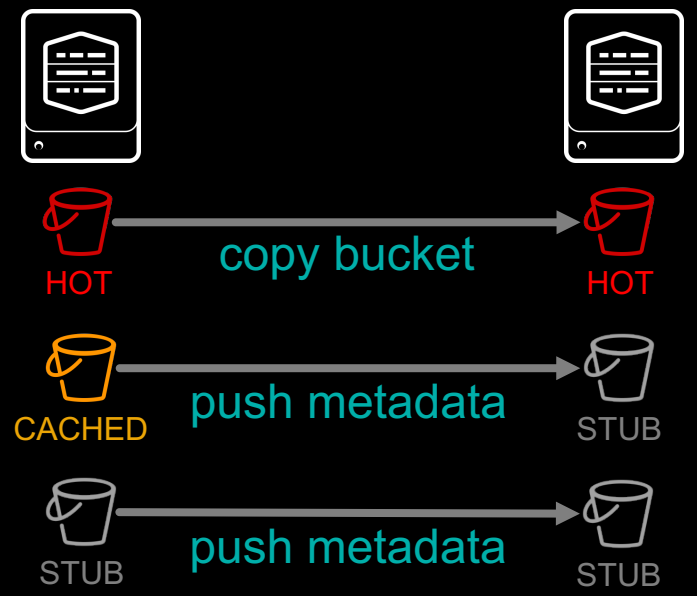
- Search Resiliency

Will my search results be complete if something goes wrong?

  - Indexer clustering must be used to provide highly available search
    - SmartStore does not natively provide search HA

Failure Scenarios

Failed peers < replication factor



Available copies of a bucket exist in the cluster

- Cluster master initiates a fix-up operation
  - Peers with an existing copy of buckets are instructed to copy them to other peers until RF/SF is met
    - Hot bucket: full copy
    - Cached bucket: metadata push
      - Peers don't need the full contents of cached buckets, as it can be fetched from the remote store
      - Metadata replication is done with a POST to a REST endpoint on the target peer(s)

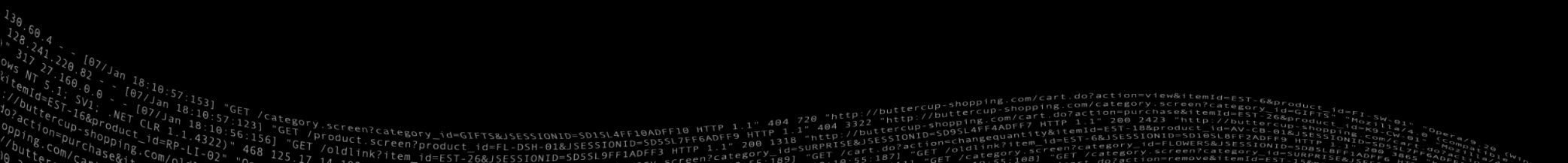
Retention

Options

Retention for SmartStore enabled indexes is managed globally

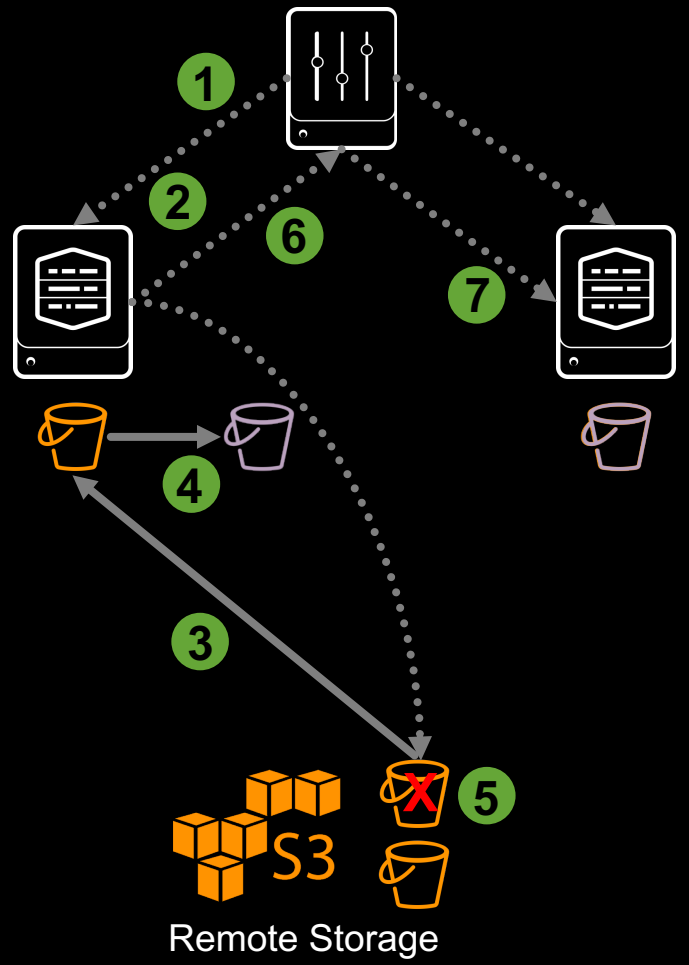
- Retention for “classic” indexes is managed on a per-indexer basis

- Retention Options
  - Total Index size in MB
    - Oldest bucket is purged when [maxGlobalDataSizeMB] is reached
  - Age of events
    - Bucket is purged when the oldest event reaches [frozenTimePeriodInSecs]



Retention

Freezing/Deleting Buckets



1. CM runs a search every 15 min on all peers to obtain a list of which buckets need to be frozen/deleted
   - [remote_storage_retention_period]
2. CM randomly assigns jobs to the peers with a copies of the buckets
3. Peer downloads the buckets it was assigned
4. Peer freezes the bucket locally
   - [coldToFrozenScript], [coldToFrozenDir], or Delete
5. Peer removes the bucket from the remote store
6. Peer notifies the CM
7. CM notifies the other peers to freeze their local copy of the bucket if they have one

Implementation

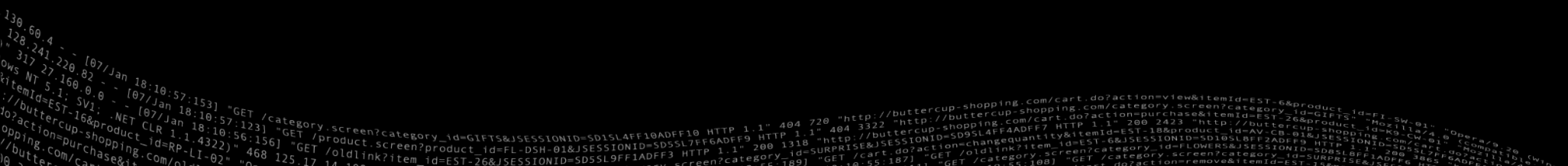
Index Configuration

- Volume Configuration

Conf File	indexes.conf
Parameters	[volume:<volume_name>] storageType = remote path = <scheme>://<remote-location-specifier> remote.s3.endpoint = <URL of S3 API> remote.s3.secret_key = remote.s3.access_key =

Conf File	indexes.conf
Parameters	[<index_name>] homePath = coldPath = *<path required, but not used>* remotePath = <volume_name>/<index_name> maxGlobalDataSizeMB = frozenTimePeriodInSecs =

- Enabled Per-Index
  - You can have a mix of SmartStore and classic indexes on the same indexer



splunk> .conf18

Customer Story

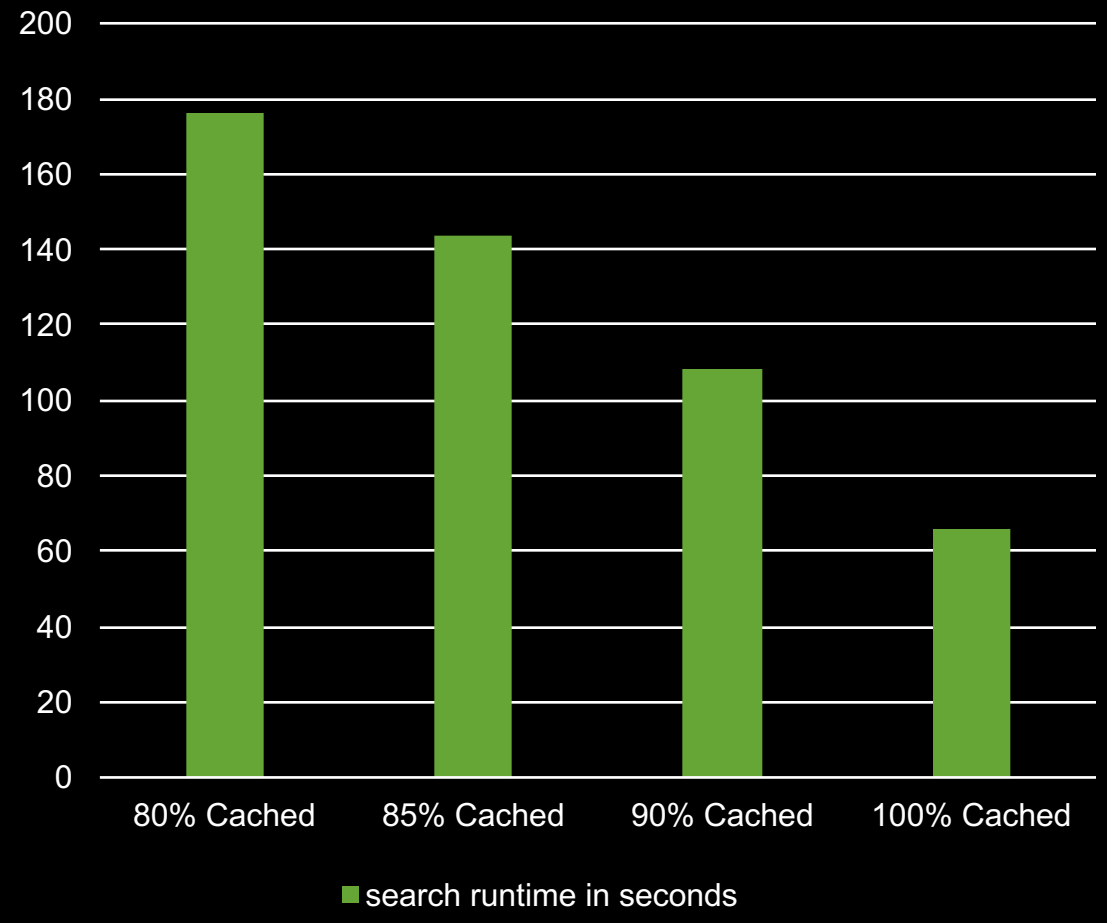
Sharing real world experience

splunk> .conf18

SmartStore Test Environment

- Focused testing on our AWS environment.
- Attached a second Lambda function to our Kinesis streams to replicate Cloudwatch Logs data to SmartStore environment's HEC endpoint (~50 indexes, 300-400 GB/day).
- Created a single AWS S3 bucket to store indexes and used an EC2 instance profile to grant the indexers access to the S3 bucket.
- Initially started with C4 instances, switched to I3.
- Testing for
  - Search Performance (cached v non-cached)
  - Indexer lifecycle (rebalance and offline)

SmartStore Search Performance



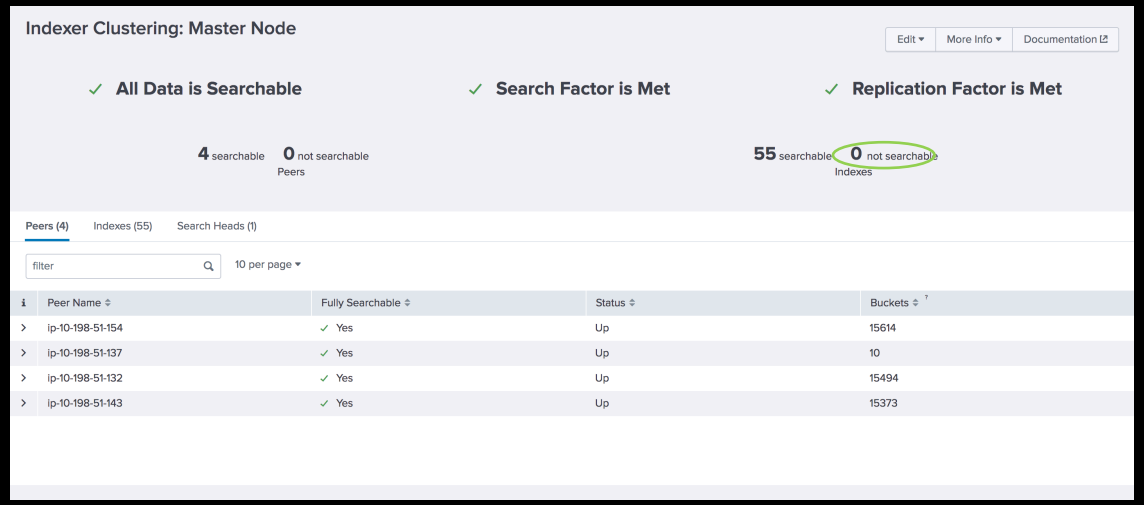
- index=foo bar earliest= -20d
- 550 buckets, ~180 GB of indexed data
- Process
  - Evicted all data from cache
  - Ran search over 5 days to warm cache by 25%, then performed search.
  - Repeated for 10 and 15 days
- Cachemgr prefetches data, testing with a completely empty cache wasn't possible

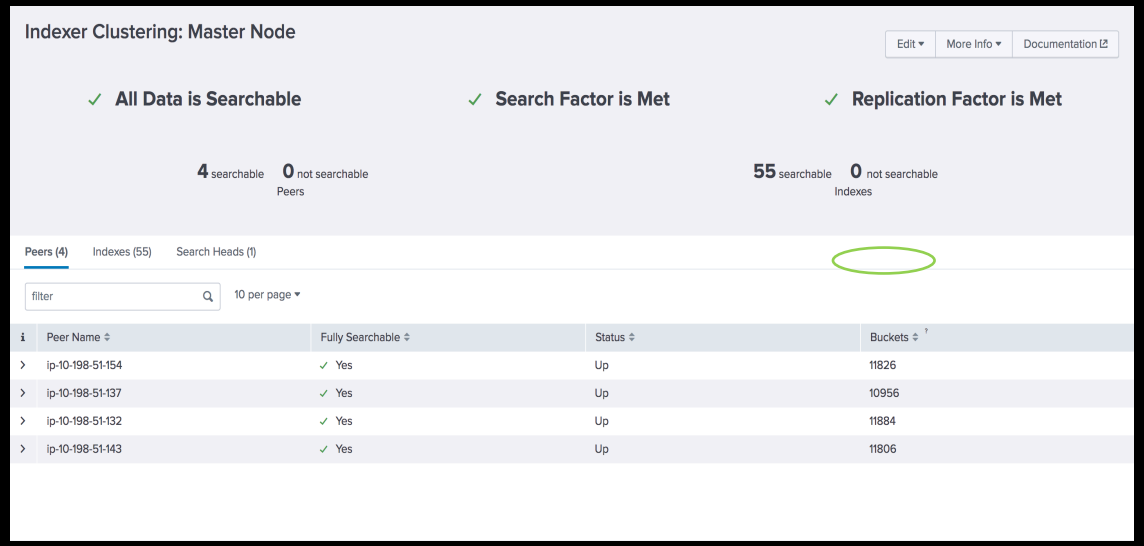
Indexer Lifecycle

Data Rebalance

Rebalancing redistributes data across an indexing cluster.

- 45k buckets (9 TB) on 3 indexers, adding 1 new indexer
- Prior to SmartStore, this process would take hours or days.
- With SmartStore, completes in less than 1 minute
- SmartStore only rebalances the buckets' pointers, so rebalancing with SmartStore is very fast.



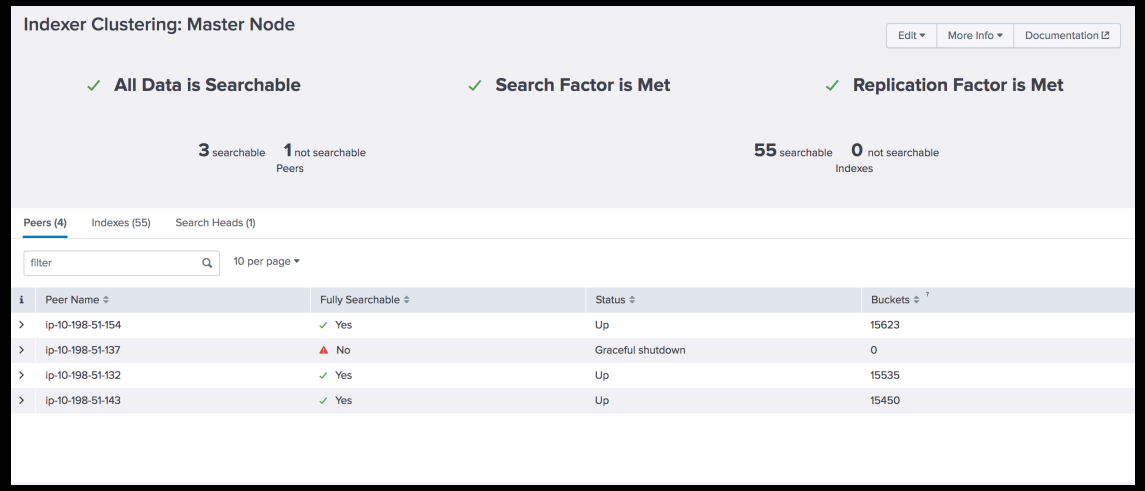


Indexer Lifecycle

Indexer/Peer Offline

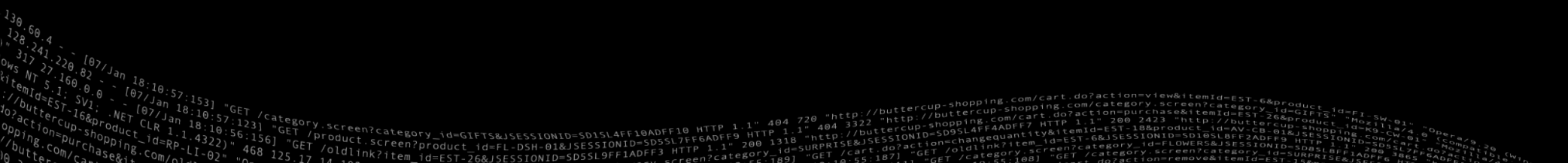
Bringing a peer node offline with enforce-counts should ensure all data is replicated before peer fully shutdown.

- Typically used to retire indexer.
- 45k buckets (9 TB) on 4 indexers, removing 1 indexer.
- Prior to SmartStore, this process could take hours or days.
- With SmartStore, completes in 8-10 minutes.



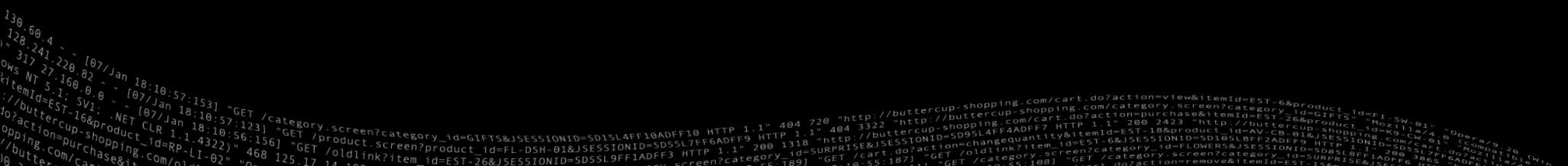
Monitoring and Troubleshooting SmartStore

- Use the SmartStore dashboard to view upload/download activity, bandwidth utilization, errors, cache hit ratio
- New _internal logging components
  - CacheManagerHandler
  - S3Client
  - HTTPClientRequest
  - CacheManager
    - action, status, download_set
- Use the search logs in _audit to show cache hits/miss/errors, can help with search/cache optimization.
- Enable AWS S3 CloudWatch Request Metrics
  - (PUT/GET/etc) request counts, errors, latency



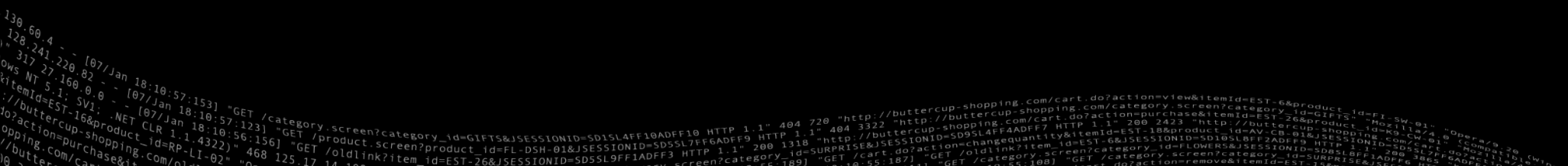
Observations

- Cache misses will slow down search results, so plan accordingly
  - Increase size of cache or qty of instances, higher bandwidth or IO instances will help
  - Given enough cache, you will rarely see misses (and is still less expensive than EBS)
  - Optimize searches that run over long periods of time
- Fast rebalance means improved indexer elasticity and recovery from failed instances
  - New peers will still need to download buckets, but the cache manager optimizes what is downloaded
  - Look at autoscaling groups with lifecycle hooks.
  - Force rebalance when you add a new peer
- Currently SmartStore is a one-way trip
  - No feature to revert index config back to non-remote
  - start with non-production environments or indexes with low retention periods



Conclusions

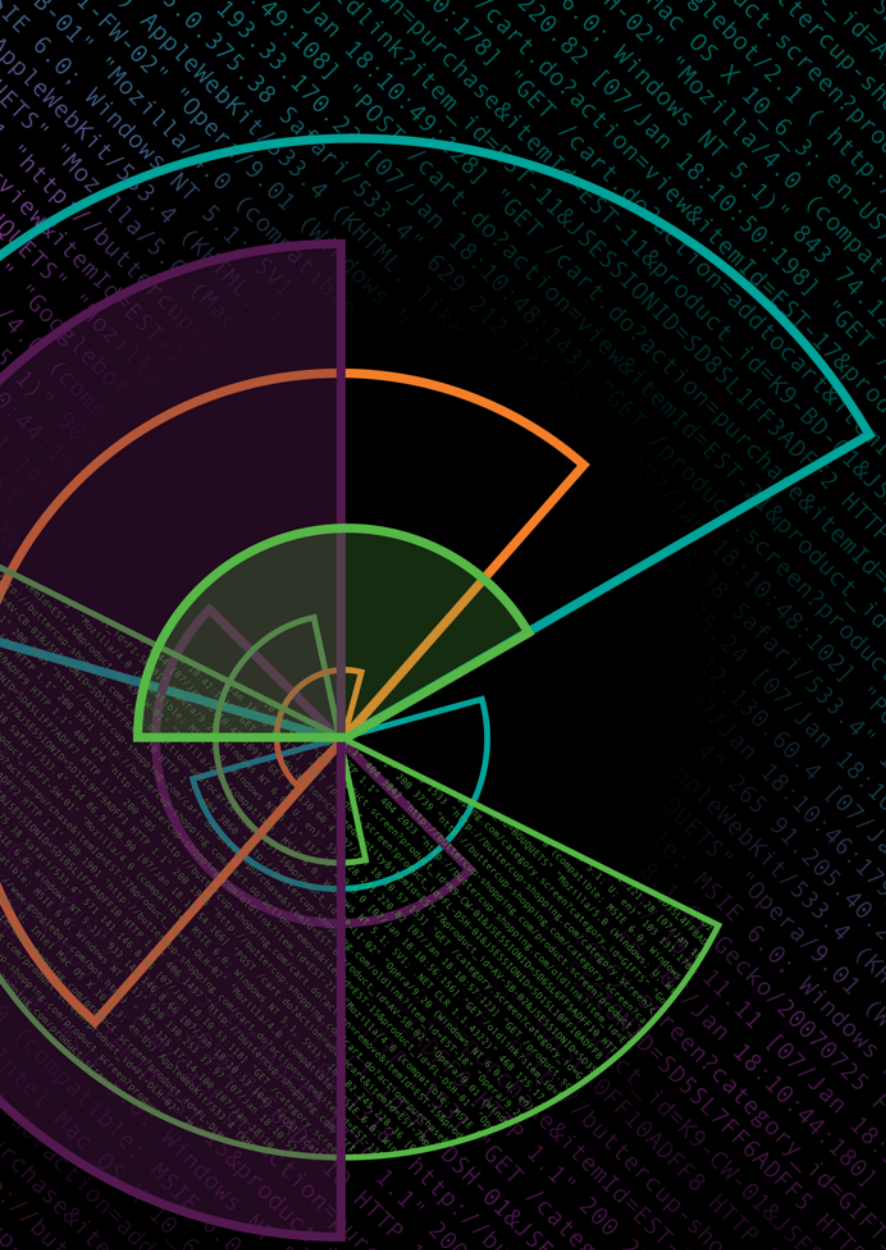
- Using SmartStore we can…
  - Increase data durability
  - Decrease AWS storage costs
  - Simplify indexer management



Splunk SmartStore

Key Takeaways

1. Splunk SmartStore is a new deployment model that scales to massive data volumes cost effectively
2. Elastically scale compute and storage on-demand
3. Significant TCO benefits and simplified cluster management



Q&A

splunk> .conf18

Thank You

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