RSA® Charge 2016

Maximizing NetWitness Performance

Sean Ennis – Principal SE, Seattle [sean.ennis@rsa.com]

#RSACharge

RSA Charge 2016

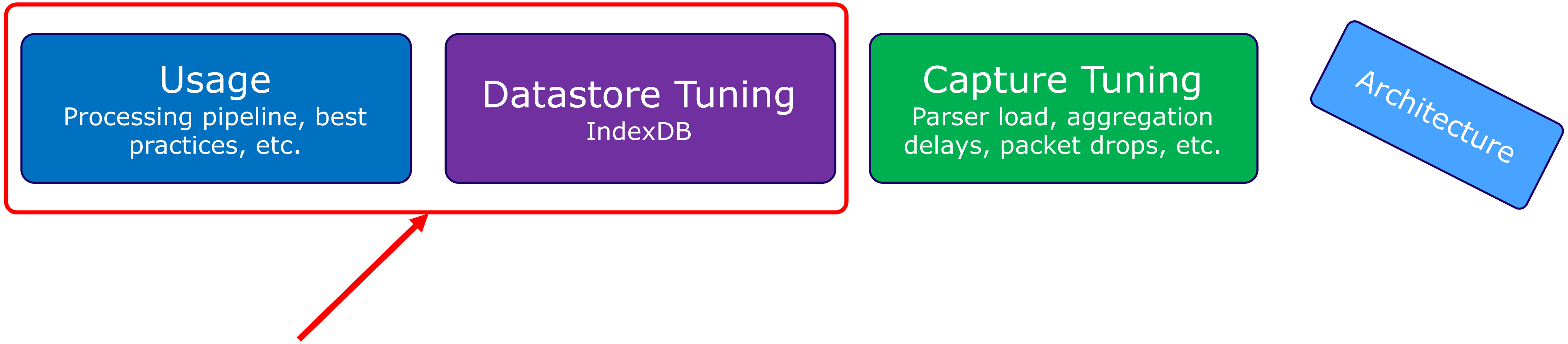
Agenda

- Overall Concept
- Optimizing Usage
  - Processing Pipeline
  - Feeds and App Rules and Lists, Oh My.
- Optimizing the Datastore (mostly index)
  - Database & Data Flow
  - Index, Index, and more Index
- Group Aggregation
- Monitoring Performance – Case Study

Overall Concept

Disclaimer: Lot's of knobs to turn, and RSA tries to minimize the requirement to do so. This presentation focuses on the most common concepts. If you are having serious performance issues, please engage your friendly RSA {SE, PS, CS} representative.

It might help to think of performance optimization in 3.5 categories:



Usage
Processing pipeline, best practices, etc.

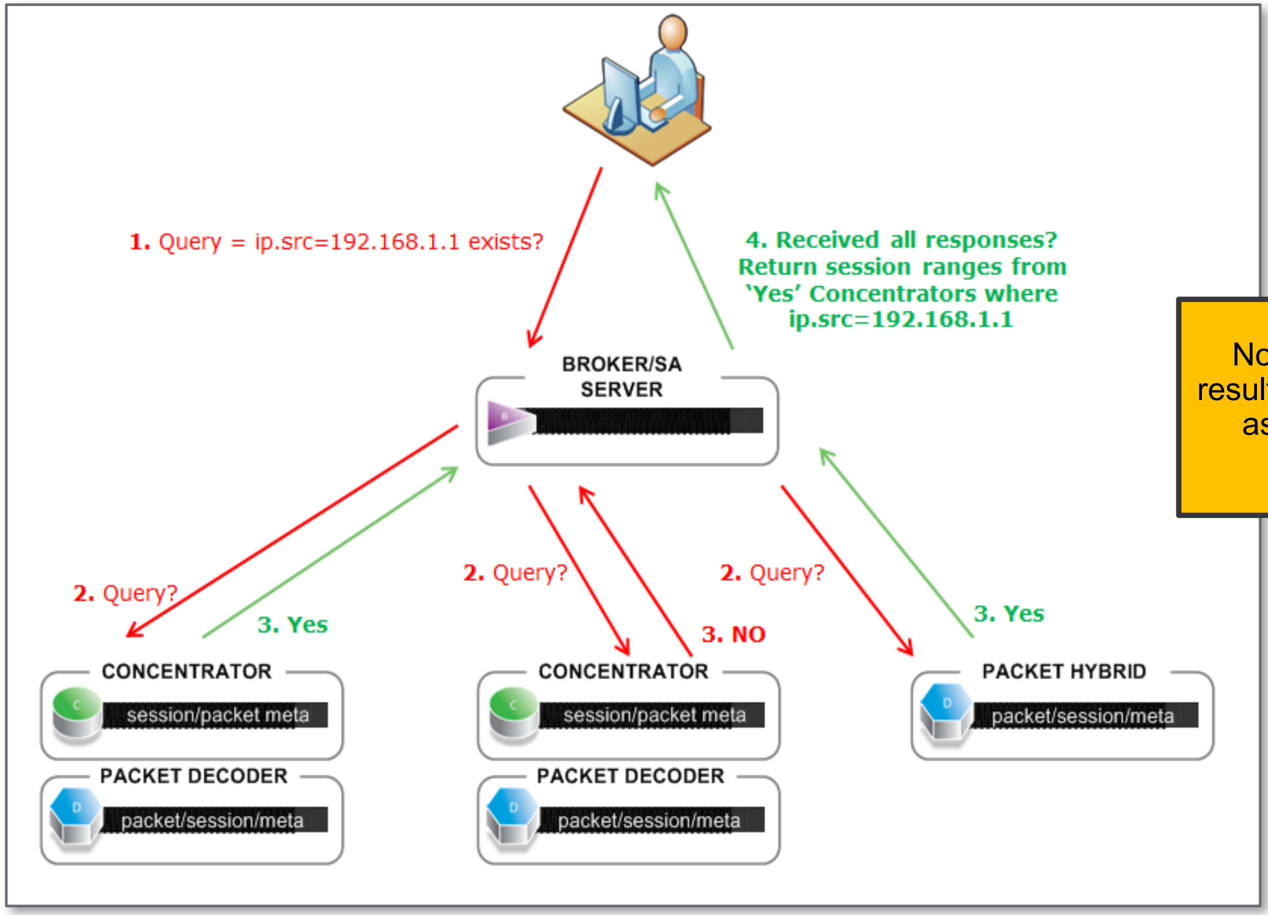
Datastore Tuning
IndexDB

Capture Tuning
Parser load, aggregation delays, packet drops, etc.

Architecture

Optimizing Usage

Query Architecture

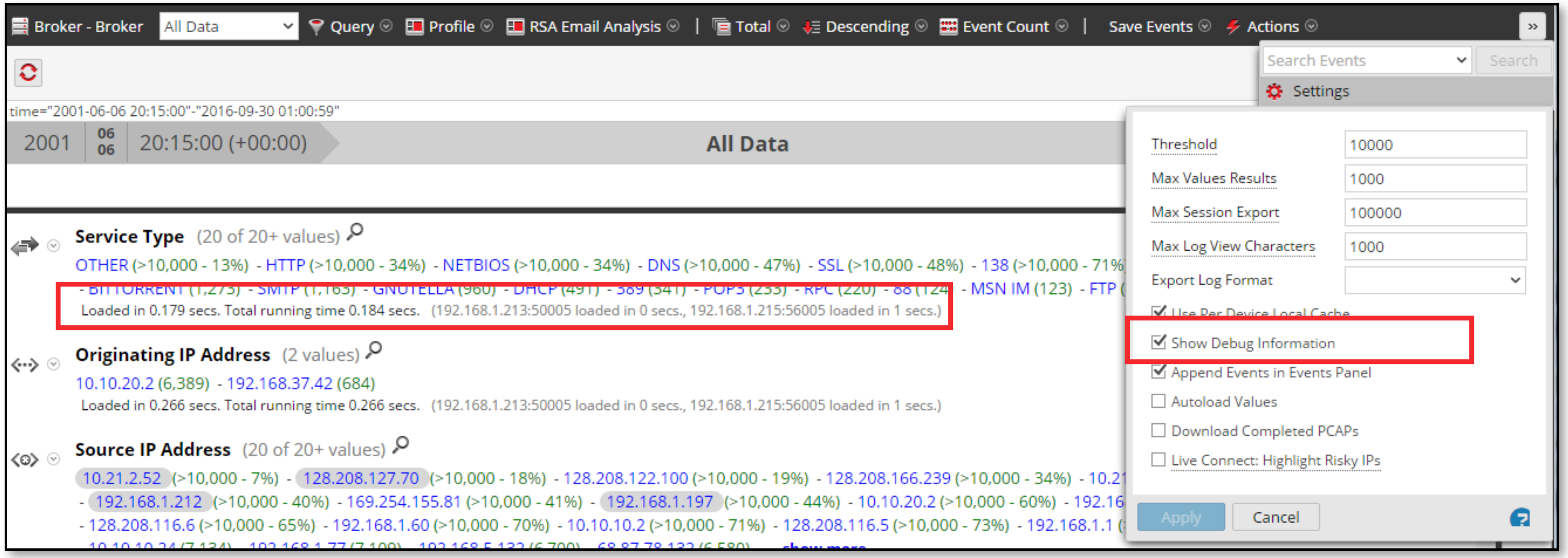


Note: 10.4+ uses "partial results", so results are loaded as they come in. Feels faster.

A query is not complete until all constituent concentrators/brokers return their results. So, 1 slow concentrator can ruin the whole party.

RSA Charge 2016

Query Architecture con’t.



time="2001-06-06 20:15:00"-"2016-09-30 01:00:59"

Service Type (20 of 20+ values)

OTHER (>10,000 - 13%) - HTTP (>10,000 - 34%) - NETBIOS (>10,000 - 34%) - DNS (>10,000 - 47%) - SSL (>10,000 - 48%) - 138 (>10,000 - 71%
- BITTORRENT (1,273) - SMTP (1,163) - GNUTELLA (960) - DHCP (491) - 389 (341) - POP3 (233) - RPC (220) - 88 (124) - MSN IM (123) - FTP (

Loaded in 0.179 secs. Total running time 0.184 secs. (192.168.1.213:50005 loaded in 0 secs., 192.168.1.215:56005 loaded in 1 secs.)

Originating IP Address (2 values)

10.10.20.2 (6,389) - 192.168.37.42 (684)

Loaded in 0.266 secs. Total running time 0.266 secs. (192.168.1.213:50005 loaded in 0 secs., 192.168.1.215:56005 loaded in 1 secs.)

Source IP Address (20 of 20+ values)

10.21.2.52 (>10,000 - 7%) - 128.208.127.70 (>10,000 - 18%) - 128.208.122.100 (>10,000 - 19%) - 128.208.166.239 (>10,000 - 34%) - 10.21
- 192.168.1.212 (>10,000 - 40%) - 169.254.155.81 (>10,000 - 41%) - 192.168.1.197 (>10,000 - 44%) - 10.10.20.2 (>10,000 - 60%) - 192.16
- 128.208.116.6 (>10,000 - 65%) - 192.168.1.60 (>10,000 - 70%) - 10.10.10.2 (>10,000 - 71%) - 128.208.116.5 (>10,000 - 73%) - 192.168.1.1 (
- 10.10.10.24 (7,134) - 192.168.1.77 (7,109) - 192.168.5.132 (6,700) - 68.87.78.132 (6,580) ... show more

Threshold	10000
Max Values Results	1000
Max Session Export	100000
Max Log View Characters	1000
Export Log Format	

☑ Use Per Device Local Cache
☑ Show Debug Information
☑ Append Events in Events Panel
☐ Autoload Values
☐ Download Completed PCAPs
☐ Live Connect: Highlight Risky IPs

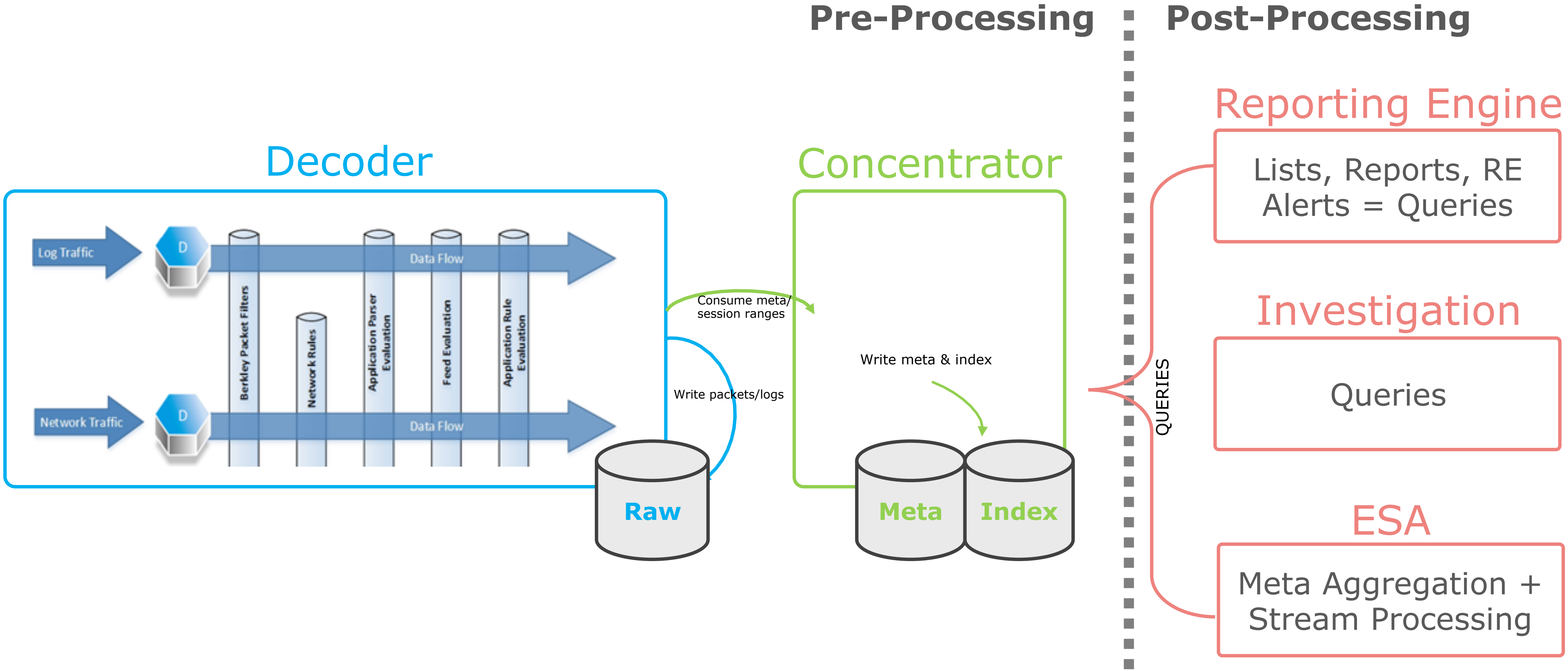
Apply | Cancel

Takeaways:

- Be aware of how many concentrators your query touches (log only? No need to query packet concentrators)
- Turn on Debug in Investigation
- In multi-site/large environments, consider Brokers to break up into queryable groups

Processing Pipeline

(slightly simplified)



Pre-Processing

Post-Processing

Decoder

Raw

Write packets/logs

Consume meta/ session ranges

Concentrator

Write meta & index

Meta

Index

QUERIES

Reporting Engine

Lists, Reports, RE Alerts = Queries

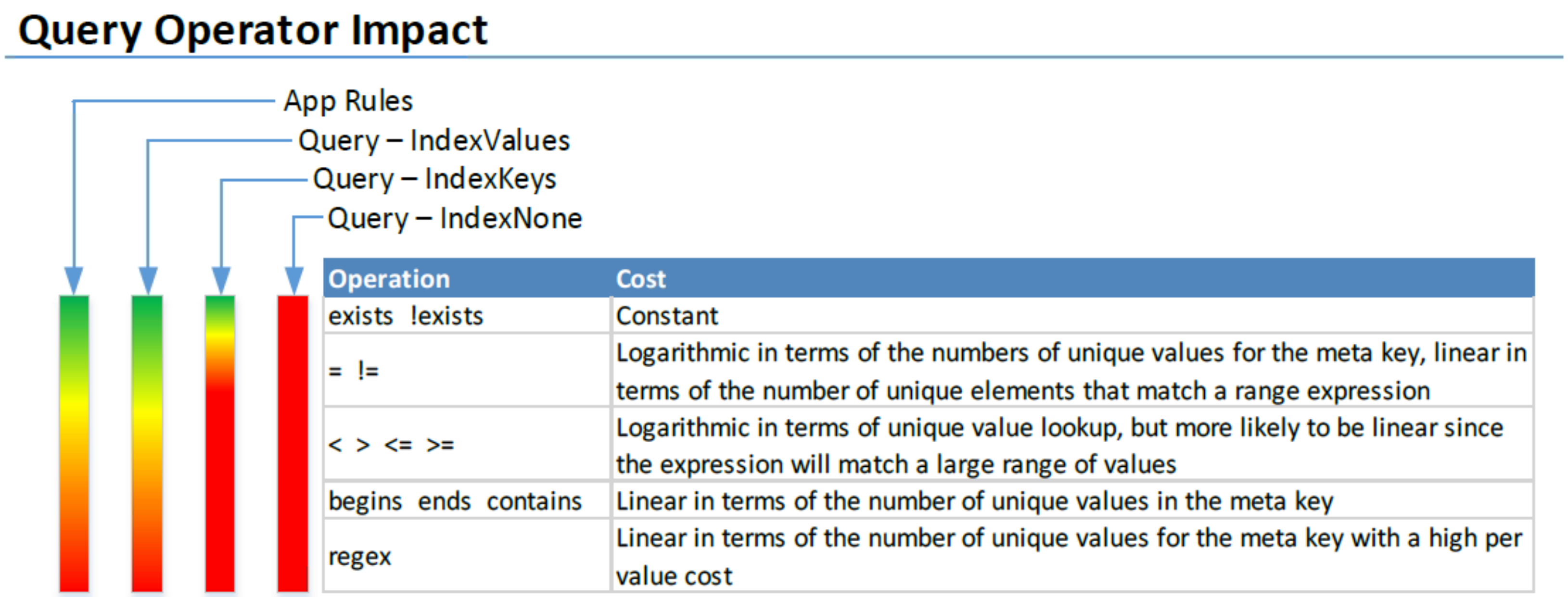
Investigation

Queries

ESA

Meta Aggregation + Stream Processing

Operator Impact



Query Operator Impact

App Rules

Query – IndexValues

Query – IndexKeys

Query – IndexNone

Operation	Cost
exists !exists	Constant
= !=	Logarithmic in terms of the numbers of unique values for the meta key, linear in terms of the number of unique elements that match a range expression
< > <= >=	Logarithmic in terms of unique value lookup, but more likely to be linear since the expression will match a large range of values
begins ends contains	Linear in terms of the number of unique values in the meta key
regex	Linear in terms of the number of unique values for the meta key with a high per value cost

Note: Optimizations made in 10.5 to underlying logic engine (OR)

Takeaways:

- Move as much to “pre processing” as possible. App rules & Feeds are your best friend. Results in single keys to query.
- Use feeds instead of Reporting Engine lists whenever possible (RE Lists effectively break up into many logical OR statements)
- Don’t use meta groups with ALL keys open. Break the problem down and open the minimal number to start (every open is a query)
- Smaller, more specific meta groups.

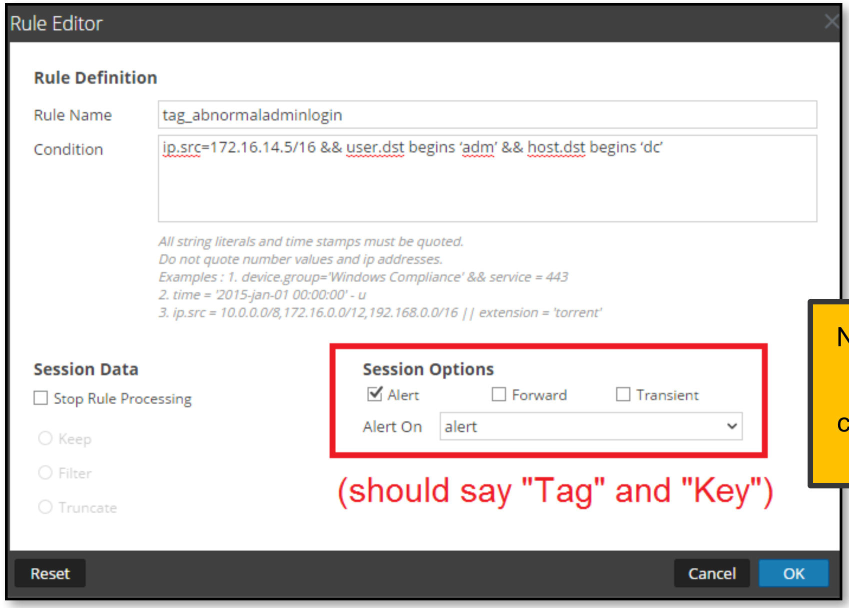
Ex. App rules

Use case: Very frequently looking for users logging in to certain hosts with admin accounts from a particular subnet.

Investigator Query (post processing) with no pre-processing app rule:

```
ip.src=172.16.14.5/16 && user.dst begins ‘adm’ && host.dst begins ‘dc’
```

Instead, what about creating an app rule to move the processing earlier in the pipeline and create a single meta value (*Admin -> Decoder -> Config -> App Rules*)?



Rule Editor

Rule Definition

Rule Name: tag_abnormaladminlogin

Condition: ip.src=172.16.14.5/16 && user.dst begins ‘adm’ && host.dst begins ‘dc’

All string literals and time stamps must be quoted.
Do not quote number values and ip addresses.
Examples : 1. device.group='Windows Compliance' && service = 443
2. time = '2015-jan-01 00:00:00' - u
3. ip.src = 10.0.0.0/8,172.16.0.0/12,192.168.0.0/16 || extension = 'torrent'

Session Data

☐ Stop Rule Processing

○ Keep
○ Filter
○ Truncate

Session Options

☑ Alert ☐ Forward ☐ Transient

Alert On: alert

(should say "Tag" and "Key")

Note: Could optimize this even further by using a FEED to track admin accounts and critical hosts. This would save logic processing.

New investigator query (or RE rule) to get the same data:

```
alert=‘tag_abnormaladminlogin’
```

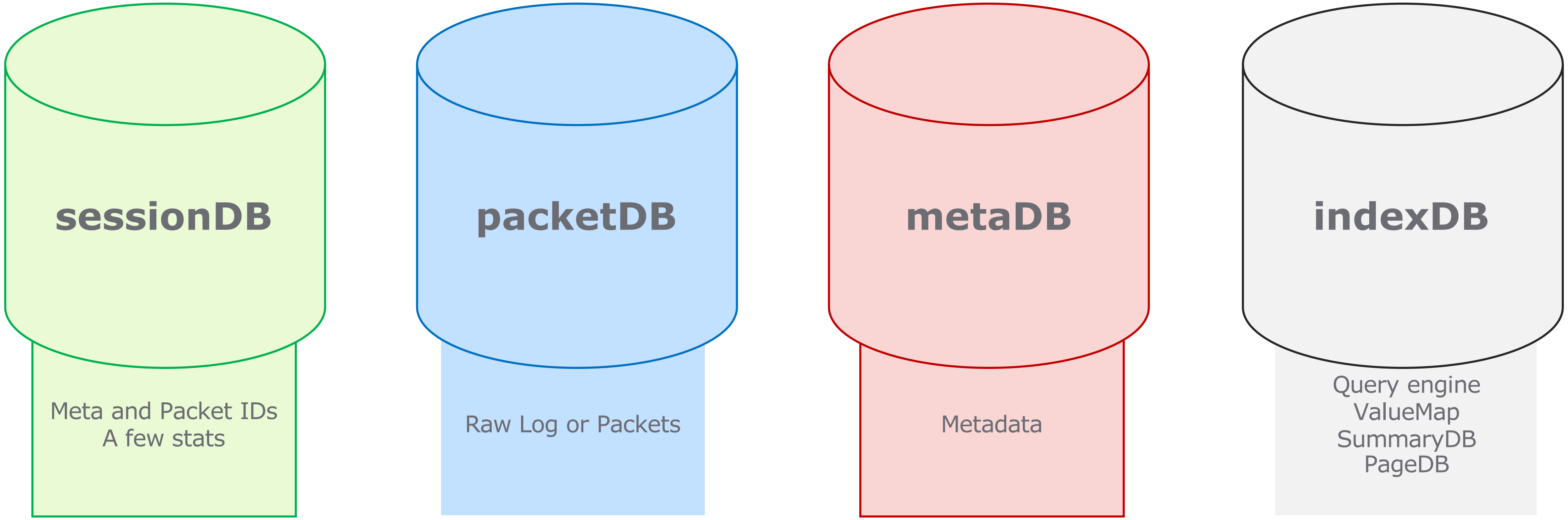
Ex. Feeds vs Lists

Use case: Daily report for traffic to/from a list of critical internal hosts

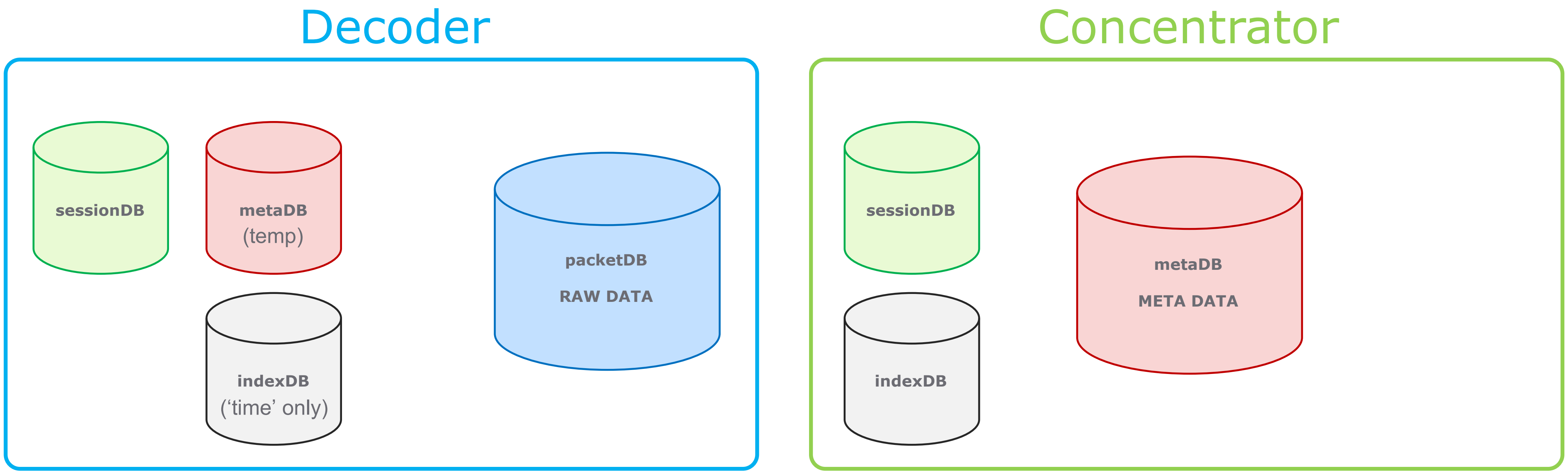
Optimizing the Datastore

Databases

Use case: Very frequently looking for users logging in to certain hosts with admin accounts from a particular subnet.



Databases by Decoder/Concentrator

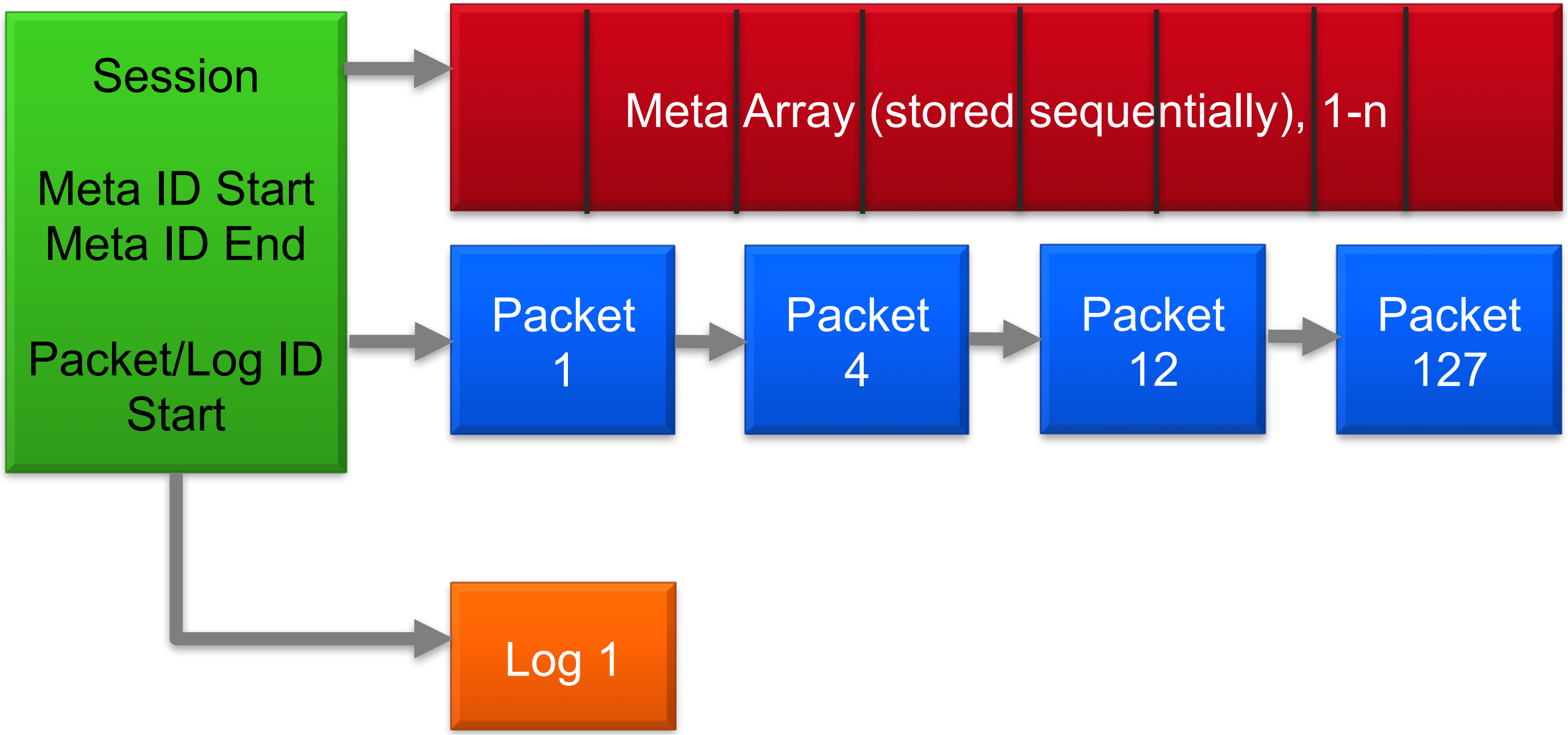


Most important for our purposes are:

packetDB(Decoder)
metaDB(Concentrator)
indexDB(Concentrator) ** *Heavily impacts performance*

RSA Charge 2016

Data Model



Session

Meta ID Start
Meta ID End

Packet/Log ID
Start

Meta Array (stored sequentially), 1-n

Packet 1

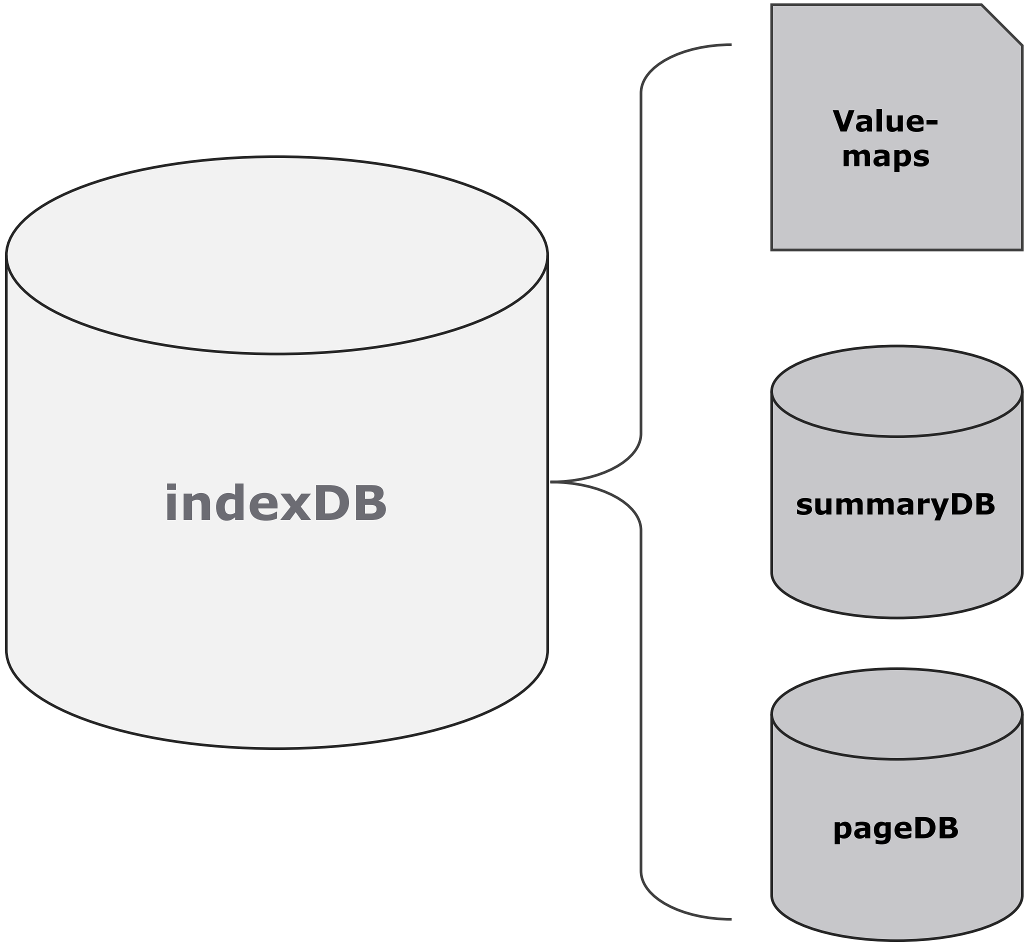
Packet 4

Packet 12

Packet 127

Log 1

Indexing – the DB

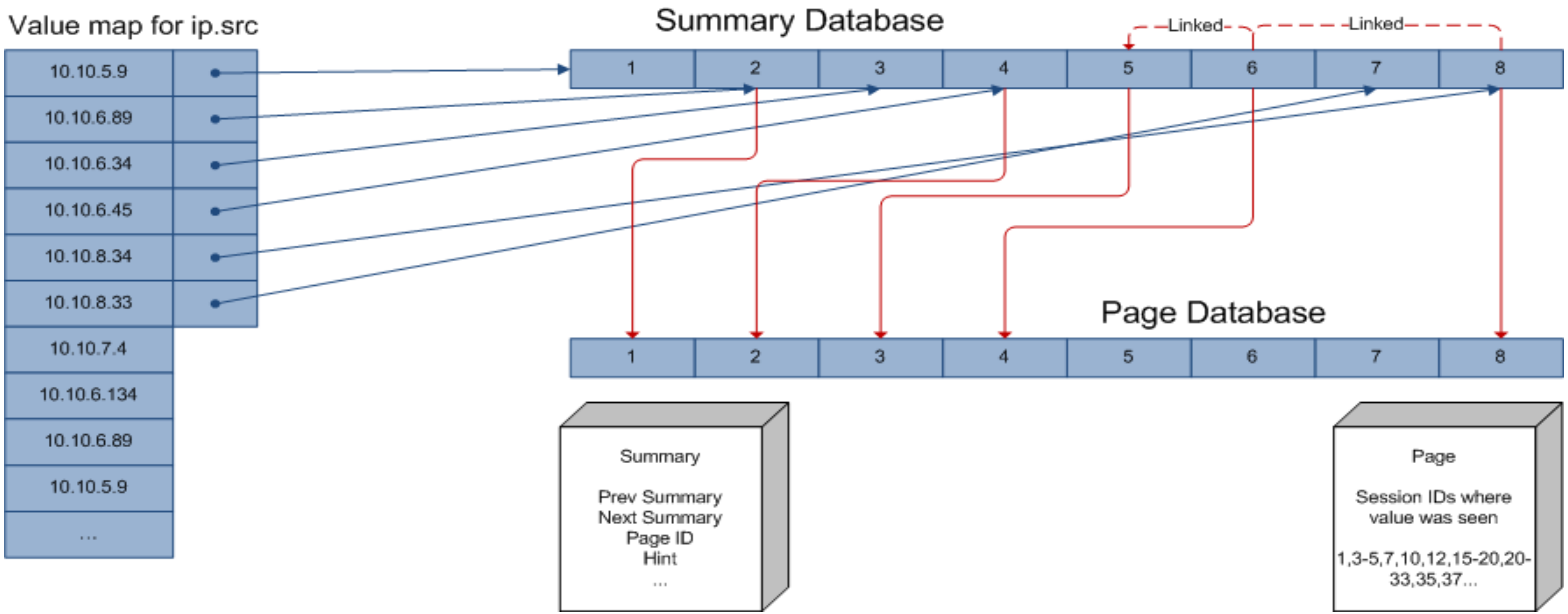


- One file per key, per slice
- Contains all unique values seen during that period (up to the defined valuesMax)
- For each value, there’s a **link to the summaryDB**

- For each unique value, contains various counts/stats and a **link to the pageDB**

- Compressed storage of session IDs
- Used to locate actual sessions containing reference to meta key/value.

Indexing – the DB, con't



Value map for ip.src

10.10.5.9

10.10.6.89

10.10.6.34

10.10.6.45

10.10.8.34

10.10.8.33

10.10.7.4

10.10.6.134

10.10.6.89

10.10.5.9

...

Summary Database

1 2 3 4 5 6 7 8

Linked

Linked

Page Database

1 2 3 4 5 6 7 8

Summary

Prev Summary
Next Summary
Page ID
Hint
...

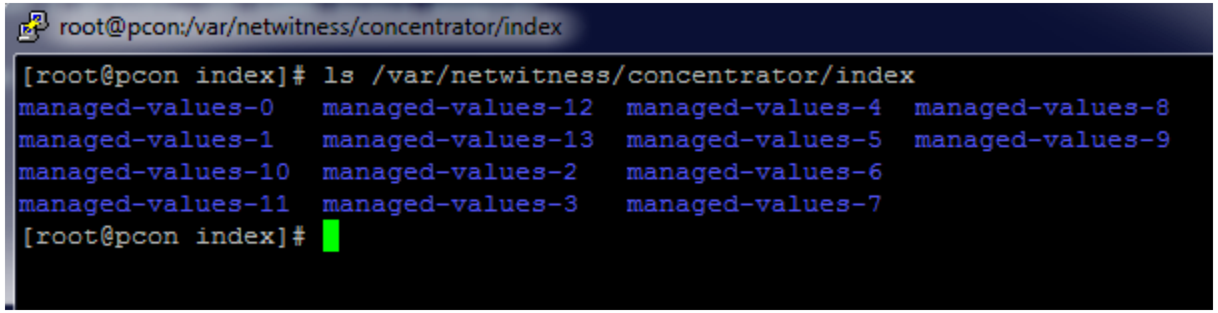
Page

Session IDs where value was seen

1,3-5,7,10,12,15-20,20-33,35,37...

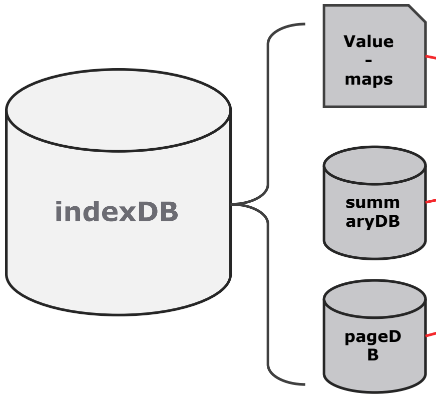
Indexing – the DB. SLICES.

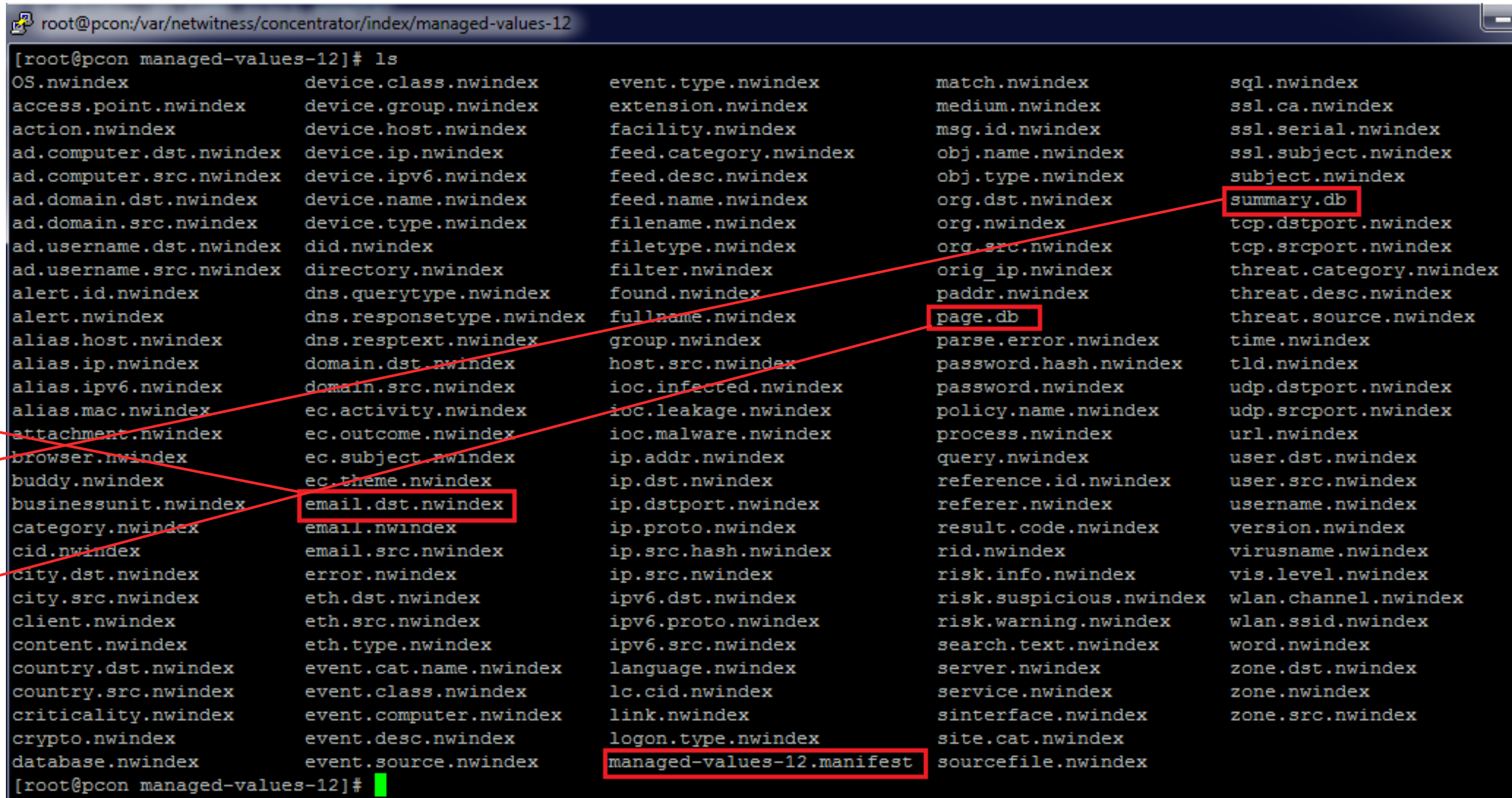
- NW holds the current slice in memory (fast) but needs to flush (save) to disk after a period of time OR number of sessions.
- Pre 10.5 = scheduled job to save every 8 hours
- Post 10.5 = save every 600,000,000 sessions
- **Note:** If upgraded through 10.5, the default 8 hour schedule persists. Fresh 10.5+ installs default to session count saves.



```
[root@pcon index]# ls /var/netwitness/concentrator/index
managed-values-0   managed-values-12  managed-values-4  managed-values-8
managed-values-1   managed-values-13  managed-values-5  managed-values-9
managed-values-10  managed-values-2   managed-values-6
managed-values-11  managed-values-3   managed-values-7
[root@pcon index]#
```

slice = /var/netwitness/concentrator/index/managed-values-X



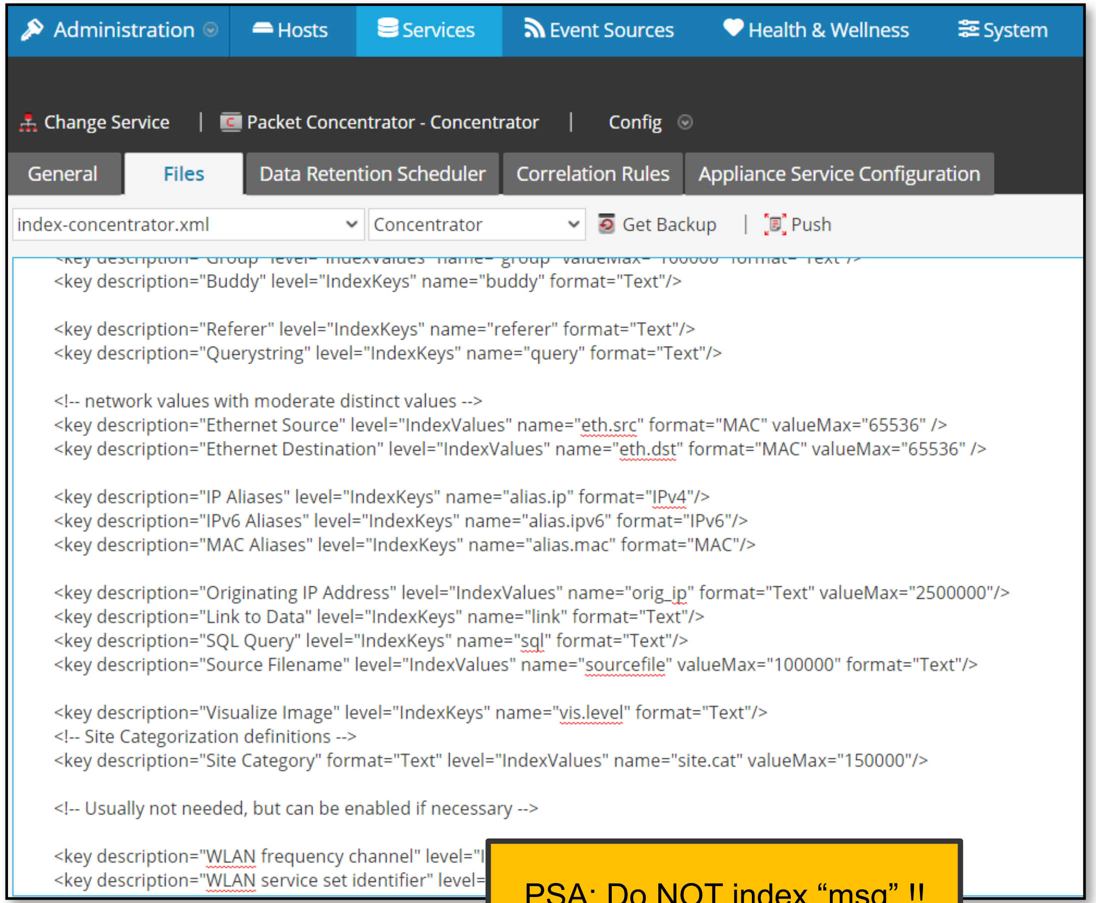


root@pcon:/var/netwitness/concentrator/index/managed-values-12

```
[root@pcon managed-values-12]# ls
OS.nwindex               device.class.nwindex      event.type.nwindex          match.nwindex            sql.nwindex
access.point.nwindex     device.group.nwindex      extension.nwindex           medium.nwindex           ssl.ca.nwindex
action.nwindex           device.host.nwindex       facility.nwindex            msg.id.nwindex           ssl.serial.nwindex
ad.computer.dst.nwindex  device.ip.nwindex         feed.category.nwindex       obj.name.nwindex         ssl.subject.nwindex
ad.computer.src.nwindex  device.ipv6.nwindex       feed.desc.nwindex           obj.type.nwindex         subject.nwindex
ad.domain.dst.nwindex    device.name.nwindex       feed.name.nwindex           org.dst.nwindex          summary.db
ad.domain.src.nwindex    device.type.nwindex       filename.nwindex            org.nwindex              tcp.dstport.nwindex
ad.username.dst.nwindex  did.nwindex               filetype.nwindex            org.src.nwindex          tcp.srcport.nwindex
ad.username.src.nwindex  directory.nwindex         filter.nwindex              orig_ip.nwindex          threat.category.nwindex
alert.id.nwindex         dns.querytype.nwindex     found.nwindex               paddr.nwindex            threat.desc.nwindex
alert.nwindex            dns.responsetype.nwindex  fullname.nwindex            page.db                  threat.source.nwindex
alias.host.nwindex       dns.resptext.nwindex      group.nwindex               parse.error.nwindex      time.nwindex
alias.ip.nwindex         domain.dst.nwindex        host.src.nwindex            password.hash.nwindex    tld.nwindex
alias.ipv6.nwindex       domain.src.nwindex        ioc.infected.nwindex        password.nwindex         udp.dstport.nwindex
alias.mac.nwindex        ec.activity.nwindex       ioc.leakage.nwindex         policy.name.nwindex      udp.srcport.nwindex
attachment.nwindex       ec.outcome.nwindex        ioc.malware.nwindex         process.nwindex          url.nwindex
browser.nwindex          ec.subject.nwindex        ip.addr.nwindex             query.nwindex            user.dst.nwindex
buddy.nwindex            ec.theme.nwindex          ip.dst.nwindex              reference.id.nwindex     user.src.nwindex
businessunit.nwindex     email.dst.nwindex         ip.dstport.nwindex          referer.nwindex          username.nwindex
category.nwindex         email.nwindex             ip.proto.nwindex            result.code.nwindex      version.nwindex
cid.nwindex              email.src.nwindex         ip.src.hash.nwindex         rid.nwindex              virusname.nwindex
city.dst.nwindex         error.nwindex             ip.src.nwindex              risk.info.nwindex        vis.level.nwindex
city.src.nwindex         eth.dst.nwindex           ipv6.dst.nwindex            risk.suspicious.nwindex  wlan.channel.nwindex
client.nwindex           eth.src.nwindex           ipv6.proto.nwindex          risk.warning.nwindex     wlan.ssid.nwindex
content.nwindex          eth.type.nwindex          ipv6.src.nwindex            search.text.nwindex      word.nwindex
country.dst.nwindex      event.cat.name.nwindex    language.nwindex            server.nwindex           zone.dst.nwindex
country.src.nwindex      event.class.nwindex       lc.cid.nwindex              service.nwindex          zone.nwindex
criticality.nwindex      event.computer.nwindex    link.nwindex                sinterface.nwindex       zone.src.nwindex
crypto.nwindex           event.desc.nwindex        logon.type.nwindex          site.cat.nwindex
database.nwindex         event.source.nwindex      managed-values-12.manifest  sourcefile.nwindex
[root@pcon managed-values-12]#
```

Indexing – Optimizations

(1) Index at the right level



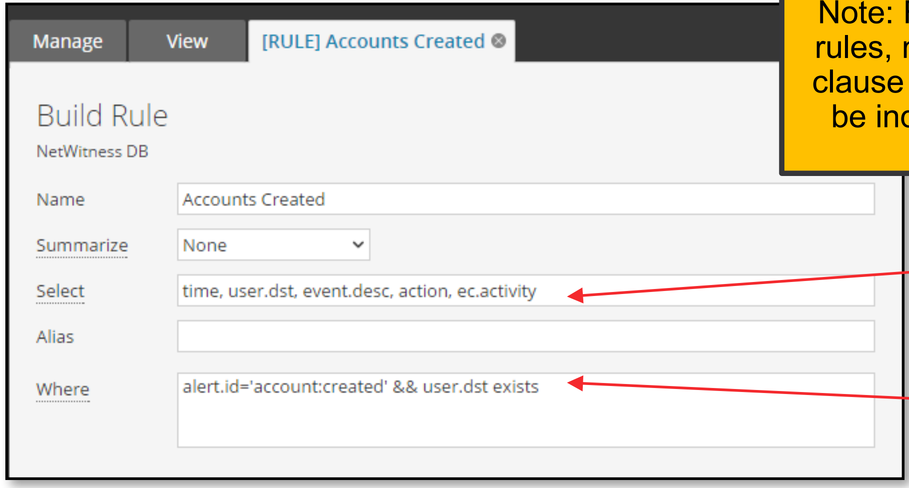
```
    <key description="Group" level="IndexValues" name="group" valueMax="100000" format="Text"/>
    <key description="Buddy" level="IndexKeys" name="buddy" format="Text"/>

    <key description="Referer" level="IndexKeys" name="referer" format="Text"/>
    <key description="Querystring" level="IndexKeys" name="query" format="Text"/>

    <!-- network values with moderate distinct values -->
    <key description="Ethernet Source" level="IndexValues" name="eth.src" format="MAC" valueMax="65536" />
    <key description="Ethernet Destination" level="IndexValues" name="eth.dst" format="MAC" valueMax="65536" />

    <key description="IP Aliases" level="IndexKeys" name="alias.ip" format="IPv4"/>
    <key description="IPv6 Aliases" level="IndexKeys" name="alias.ipv6" format="IPv6"/>
    <key description="MAC Aliases" level="IndexKeys" name="alias.mac" format="MAC"/>

    <key description="Originating IP Address" level="IndexValues" name="orig_ip" format="Text" valueMax="2500000"/>
    <key description="Link to Data" level="IndexKeys" name="link" format="Text"/>
    <key description="SQL Query" level="IndexKeys" name="sql" format="Text"/>
    <key description="Source Filename" level="IndexValues" name="sourcefile" valueMax="100000" format="Text"/>

    <key description="Visualize Image" level="IndexKeys" name="vis.level" format="Text"/>
    <!-- Site Categorization definitions -->
    <key description="Site Category" format="Text" level="IndexValues" name="site.cat" valueMax="150000"/>

    <!-- Usually not needed, but can be enabled if necessary -->

    <key description="WLAN frequency channel" level="I
    <key description="WLAN service set identifier" level=
```

PSA: Do NOT index “msg” !!

IndexKeys: Optimized for *exists/!exists* condition
IndexValues: Optimized for search/comparisons of actual values
IndexNone: Key defined, but no index

If a key needs to be searched often, you likely need IndexValues.

In investigator, you can still manually query values where index level = IndexKeys but it will be SLOW.



Name	Accounts Created
Summarize	None
Select	time, user.dst, event.desc, action, ec.activity
Alias	
Where	alert.id='account:created' && user.dst exists

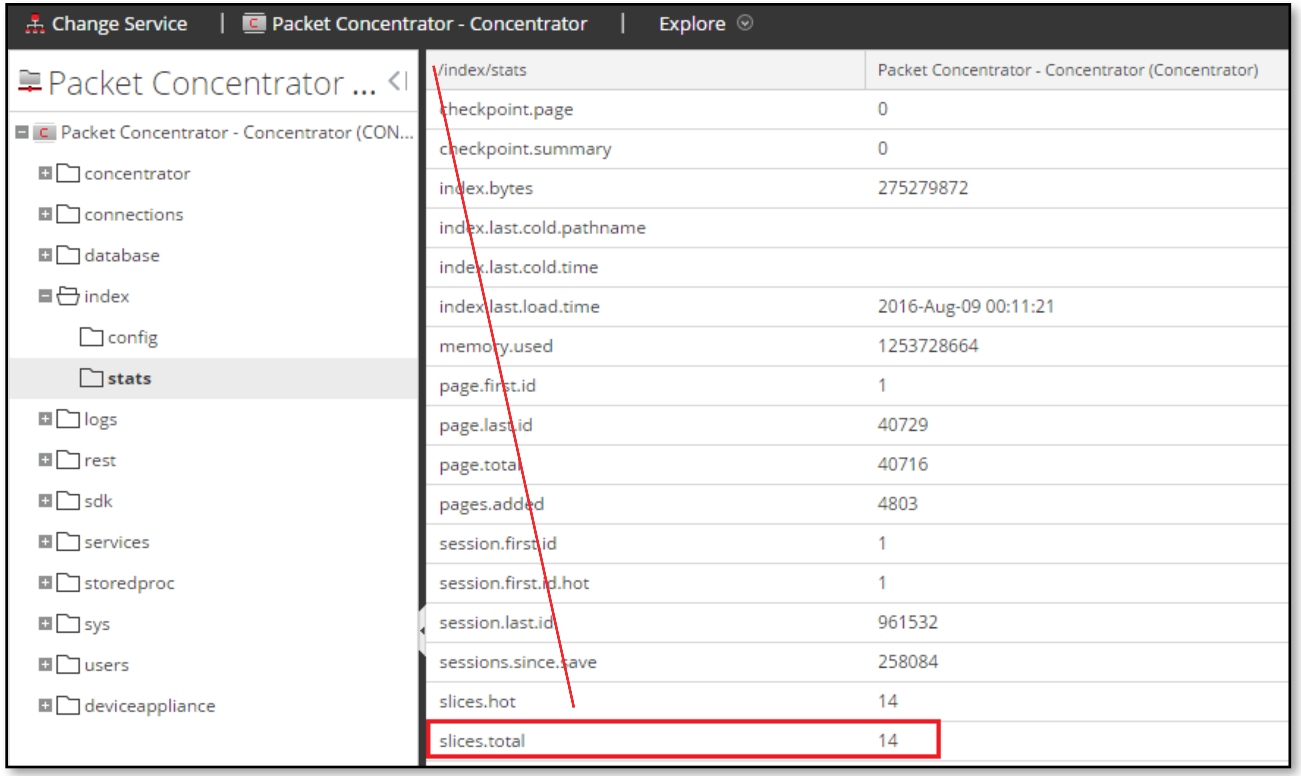
Note: For Reporting Engine rules, meta in the “WHERE” clause (not “SELECT”) must be indexed at some level.

does not need to be indexed

must be indexed

Indexing – Optimizations

(2) Keep the slice count LOW (~200-300?)

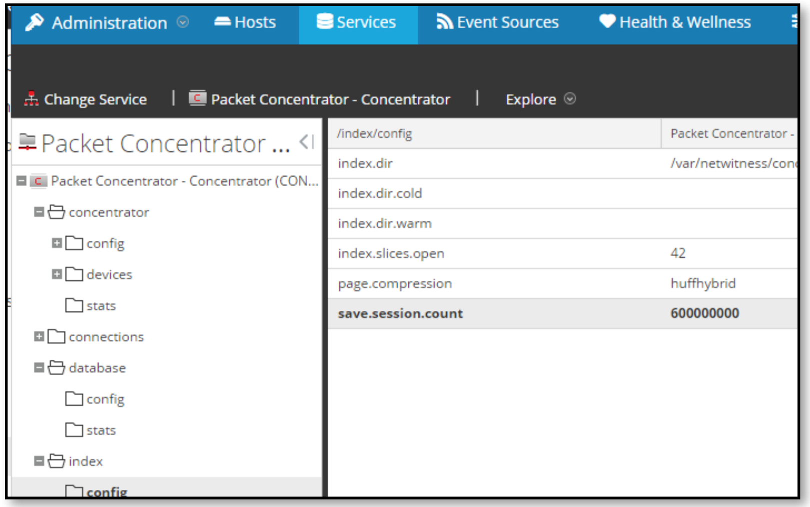


/index/stats	Packet Concentrator - Concentrator (Concentrator)
checkpoint.page	0
checkpoint.summary	0
index.bytes	275279872
index.last.cold.pathname	
index.last.cold.time	
index.last.load.time	2016-Aug-09 00:11:21
memory.used	1253728664
page.first.id	1
page.last.id	40729
page.total	40716
pages.added	4803
session.first.id	1
session.first.id.hot	1
session.last.id	961532
sessions.since.save	258084
slices.hot	14
slices.total	14

Any low volume devices initially installed @ 10.5 or earlier?

- 1 slice ever 8 hours. 300 days of metadata = ~1000 slices = SLOW.

- Install >= 10.6, defaults to 600M slices instead of time.
- Install <= 10.5, defaults to 8 hours – must change setting & remove scheduled job (***concentrator -> files -> scheduler***).



/index/config	Packet Concentrator -
index.dir	/var/netwitness/conc
index.dir.cold	
index.dir.warm	
index.slices.open	42
page.compression	huffhybrid
save.session.count	**600000000**

Indexing – Optimizations

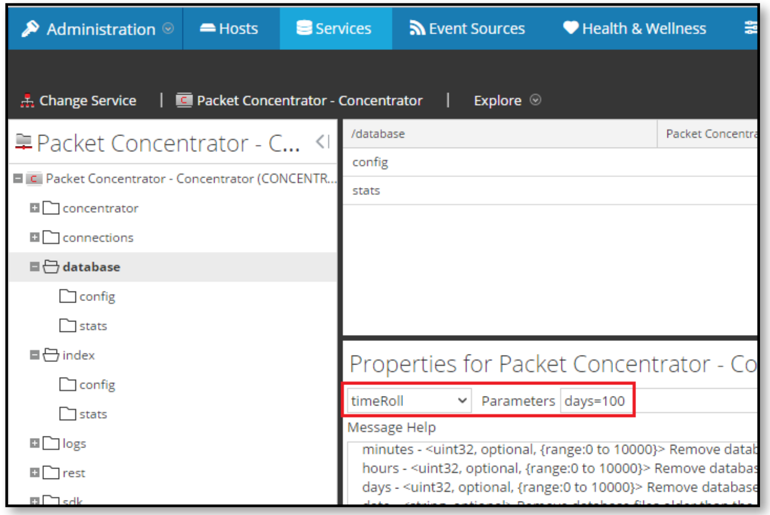
(2) Keep the slice count LOW (~200-300?) (con't)

What can you do if slice count = high?

1) Age out data for low volume devices if you can.
   Timeroll on metaDB will truncate the index on 10.5+ after next index save point.

2) Orphaned slices? Open a support ticket - delete the files.

3) >= 10.6, make sure slicing is configured by session count and
Remove time-based slice save schedule.



Properties for Packet Concentrator - Co

timeRoll Parameters days=100

Message Help

```
minutes - <uint32, optional, {range:0 to 10000}> Remove datab
hours - <uint32, optional, {range:0 to 10000}> Remove databas
days - <uint32, optional, {range:0 to 10000}> Remove database
```

Indexing – Optimizations

(3) # unique values per key, per slice < valueMax

If # unique values for a key in a slice > configured valueMax, that value becomes unsearchable.

```
<key description=“ACME Location" format="Text" level="IndexValues" name=“acme.loc" valueMax=“5“/>
```

slice1

Value	Atlanta	New York	Seattle	LA	Cleveland	Miami	Chicago
sessionIDs with value	1-5,21	6,7,50-51	8,24	11-16,18	25,27,28	29-32	45

slice2

Value	Seattle	Chicago	LA	New York	Cleveland	Atlanta	Miami
sessionIDs with value	76,77	79, 81	85-90	82, 90-92	83-84	86	99,101-103

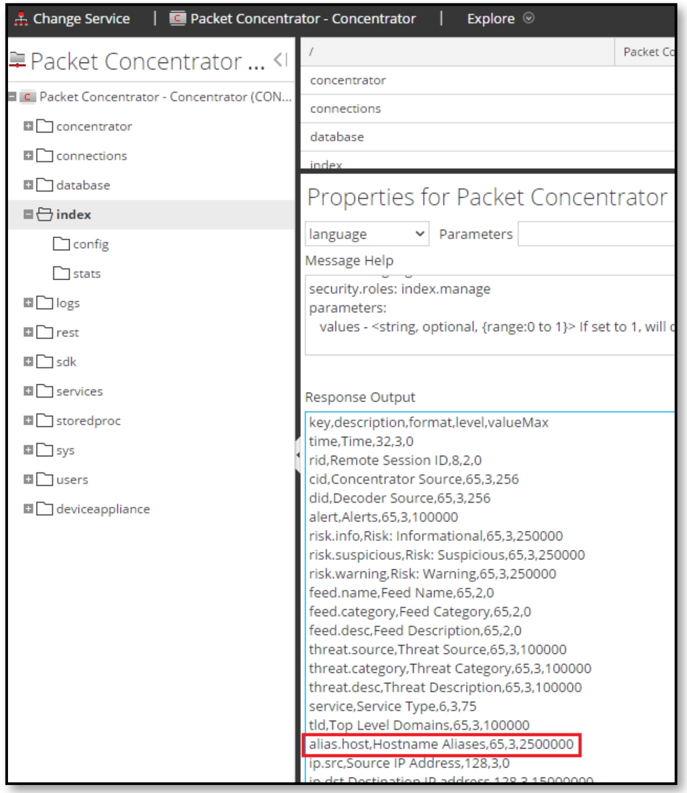
Query> acme.loc = ‘Miami’ Result Session IDs = NIL
Query> acme.loc = ‘Chicago’ Result Session IDs = 79, 81

Indexing – Optimizations

(3) # unique values per key, per slice < valueMax (con't)

So how do you check? Index inspect/language queries (API)

(1) Check config for value X

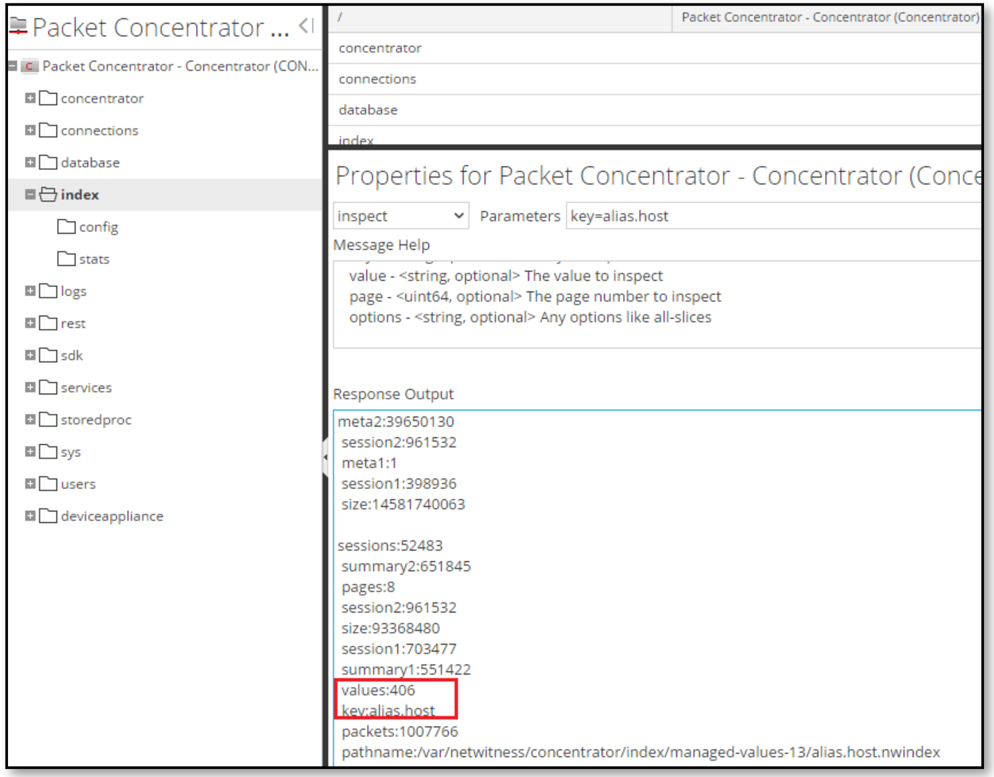


```
security.roles: index.manage
parameters:
  values - <string, optional, {range:0 to 1}> If set to 1, will c
```

```
key,description,format,level,valueMax
time,Time,32,3,0
rid,Remote Session ID,8,2,0
cid,Concentrator Source,65,3,256
did,Decoder Source,65,3,256
alert,Alerts,65,3,100000
risk.info,Risk: Informational,65,3,250000
risk.suspicious,Risk: Suspicious,65,3,250000
risk.warning,Risk: Warning,65,3,250000
feed.name,Feed Name,65,2,0
feed.category,Feed Category,65,2,0
feed.desc,Feed Description,65,2,0
threat.source,Threat Source,65,3,100000
threat.category,Threat Category,65,3,100000
threat.desc,Threat Description,65,3,100000
service,Service Type,6,3,75
tld,Top Level Domains,65,3,100000
alias.host,Hostname Aliases,65,3,2500000
ip.src,Source IP Address,128,3,0
ip.dst,Destination IP address,128,3,15000000
```

(can also check index-concentrator.xml and index-concentrator-custom.xml files)

(2) Check current slice (or all) to get # unique values for a key



Parameters: key=alias.host

```
  value - <string, optional> The value to inspect
  page - <uint64, optional> The page number to inspect
  options - <string, optional> Any options like all-slices
```

```
meta2:39650130
 session2:961532
 meta1:1
 session1:398936
 size:14581740063

sessions:52483
 summary2:651845
 pages:8
 session2:961532
 size:93368480
 session1:703477
 summary1:551422
 values:406
 key:alias.host
 packets:1007766
 pathname:/var/netwitness/concentrator/index/managed-values-13/alias.host.nwindex
```

alias.host
406/2,500,000 = **OK.**

Note: There are some user-generated scripts to automate this. Check with your local SE.

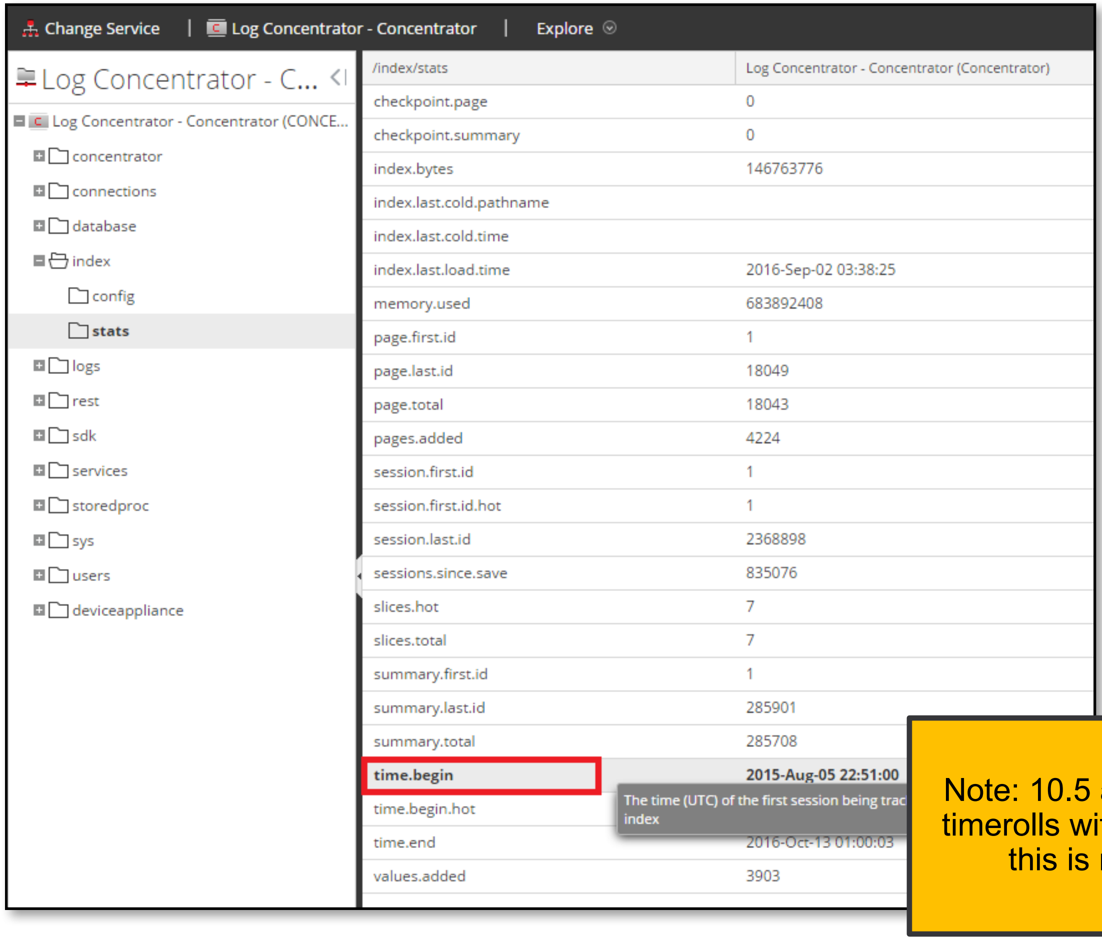
Indexing – Optimizations

(4) Index Age

Prior to 10.5, nothing cleaned up old index slices.

Result: Index Age > Meta Age (no point having an index for data that doesn’t exist)

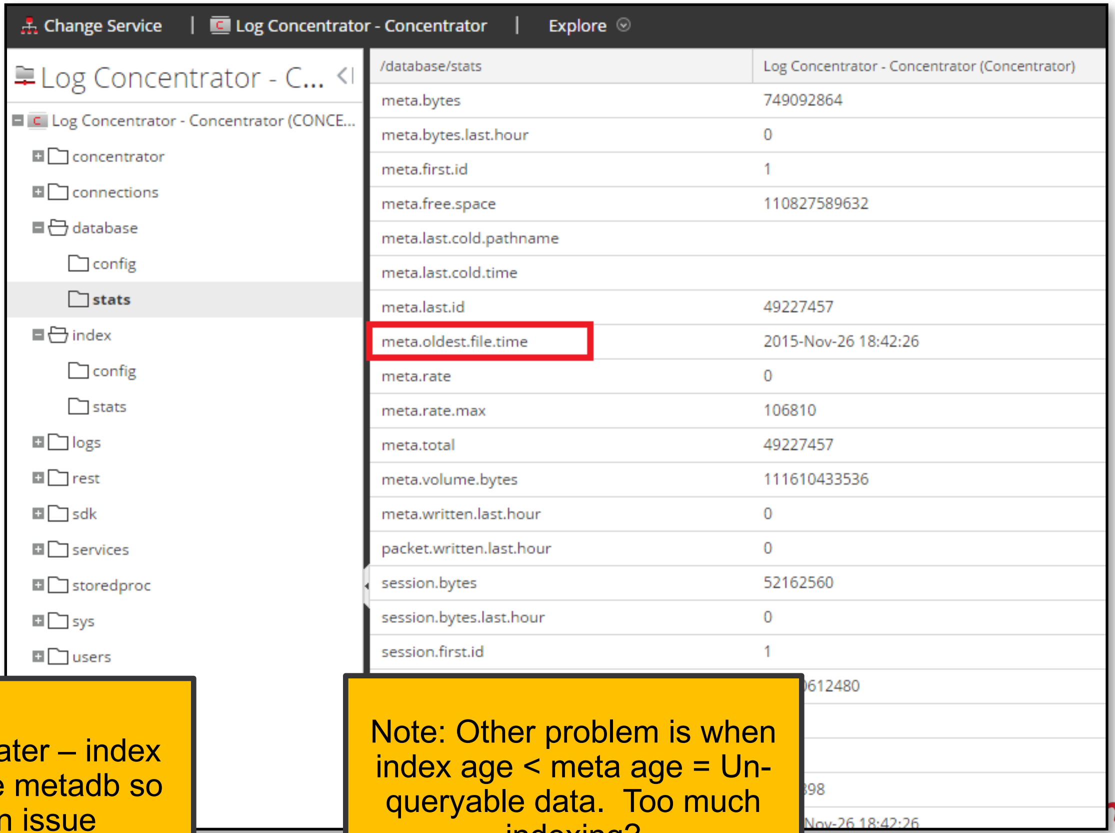
Issue: With time-based slicing, this means more slices = more overhead = slower queries.



/index/stats	Log Concentrator - Concentrator (Concentrator)
checkpoint.page	0
checkpoint.summary	0
index.bytes	146763776
index.last.cold.pathname	
index.last.cold.time	
index.last.load.time	2016-Sep-02 03:38:25
memory.used	683892408
page.first.id	1
page.last.id	18049
page.total	18043
pages.added	4224
session.first.id	1
session.first.id.hot	1
session.last.id	2368898
sessions.since.save	835076
slices.hot	7
slices.total	7
summary.first.id	1
summary.last.id	285901
summary.total	285708
time.begin	**2015-Aug-05 22:51:00**
time.begin.hot	
time.end	2016-Oct-13 01:00:03
values.added	3903

The time (UTC) of the first session being trac
index

Note: 10.5 and later – index timerolls with the metadb so this is not an issue



/database/stats	Log Concentrator - Concentrator (Concentrator)
meta.bytes	749092864
meta.bytes.last.hour	0
meta.first.id	1
meta.free.space	110827589632
meta.last.cold.pathname	
meta.last.cold.time	
meta.last.id	49227457
meta.oldest.file.time	2015-Nov-26 18:42:26
meta.rate	0
meta.rate.max	106810
meta.total	49227457
meta.volume.bytes	111610433536
meta.written.last.hour	0
packet.written.last.hour	0
session.bytes	52162560
session.bytes.last.hour	0
session.first.id	1
	0612480
	398
	Nov-26 18:42:26

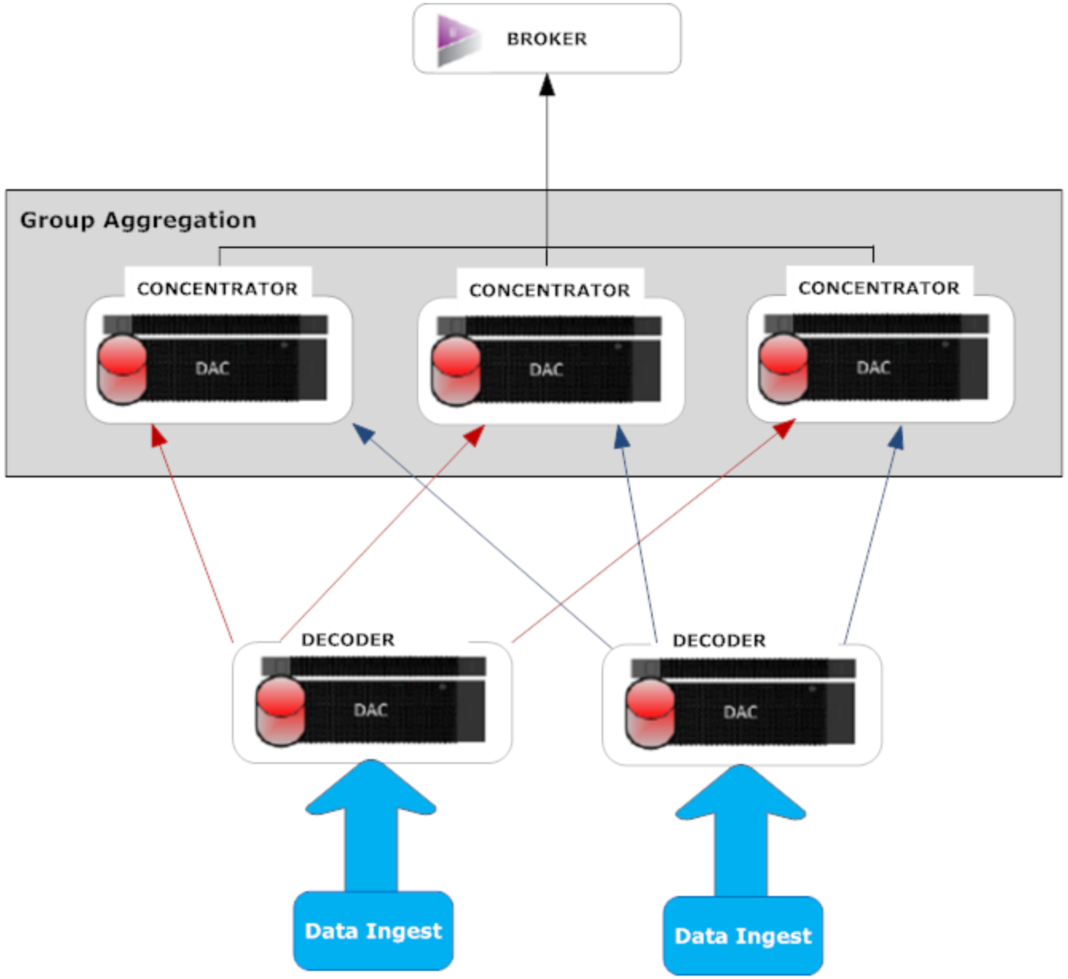
Note: Other problem is when index age < meta age = Un-queryable data. Too much indexing?

Group Aggregation

RSA Charge 2016

Group Aggregation

https://sadocs.emc.com/0_en-us/088_SA106/100_Dep/20GrpAggreg



- Effectively multiplies compute for queries
- Concentrators SPLIT the sessions between themselves (NOT HA)
- Fewer sessions per concentrator given the same amount of ingest

N:M relationship.

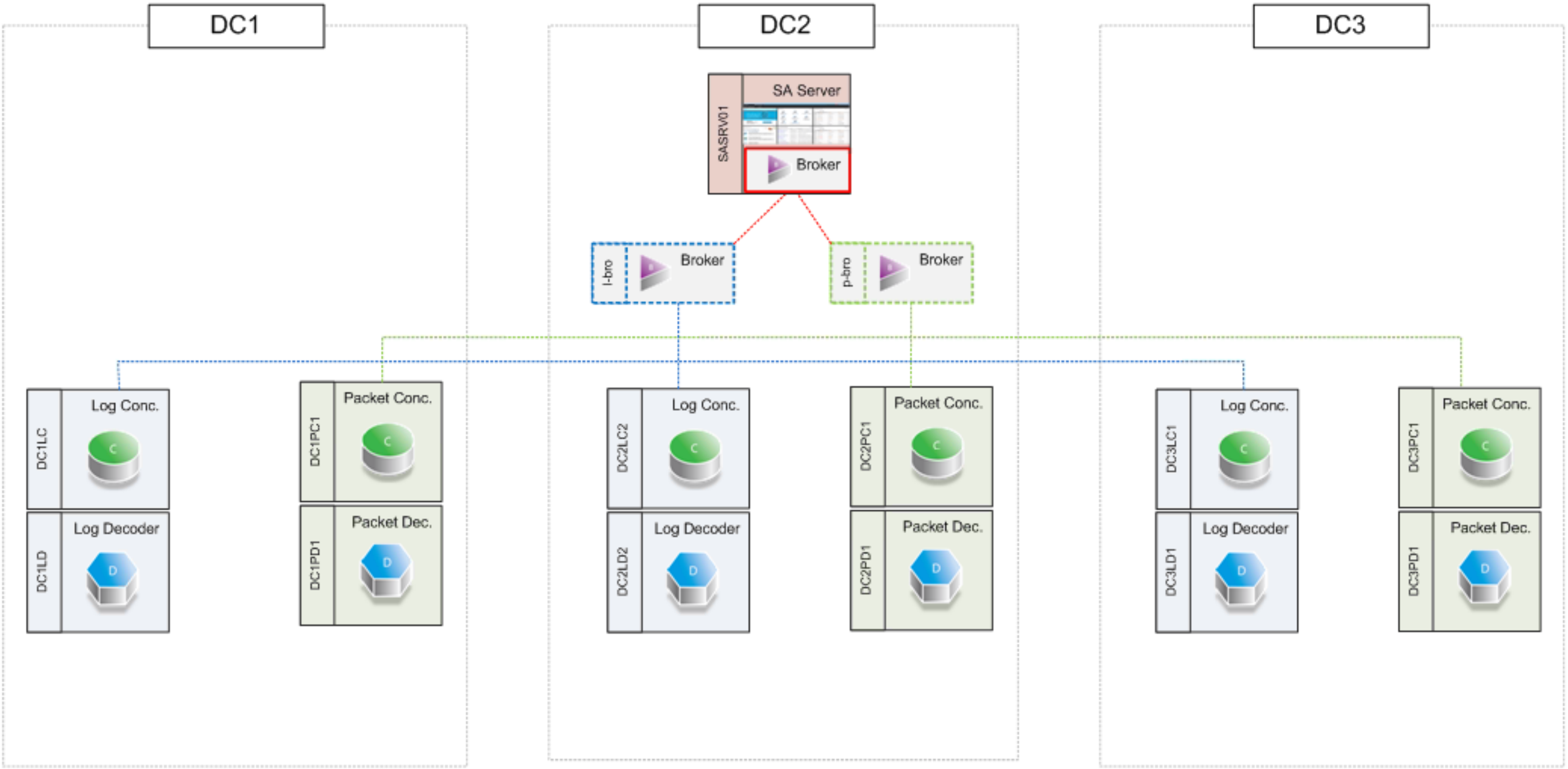
Most common group is 2 Concentrators -> 1 Decoder.

Monitoring Performance

A Real World Study

RSA Charge 2016

Case Study – Noname Inc.



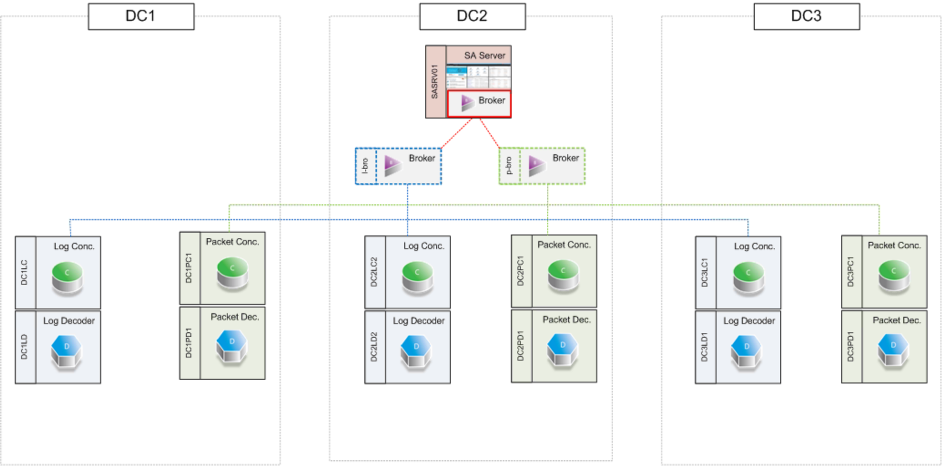
Symptoms:

1) Analysts: "We can't use the system – it's too slow"
2) Reports timing out (blank reports in the morning)
3) Inconsistent reporting against meta keys (gaps in data where certain values should exist)

3 x Log Decoder/Concentrator Stacks
3 x Packet Decoder/Concentrator Stacks
1 x Global Broker
2 x Type Broker (1 x Log, 1 x Packet)

Packet Requirements: 30 days of metadata, 7 days of raw
Log Requirements: 60 days of metadata, 60 days of raw

Case Study – Noname Inc.



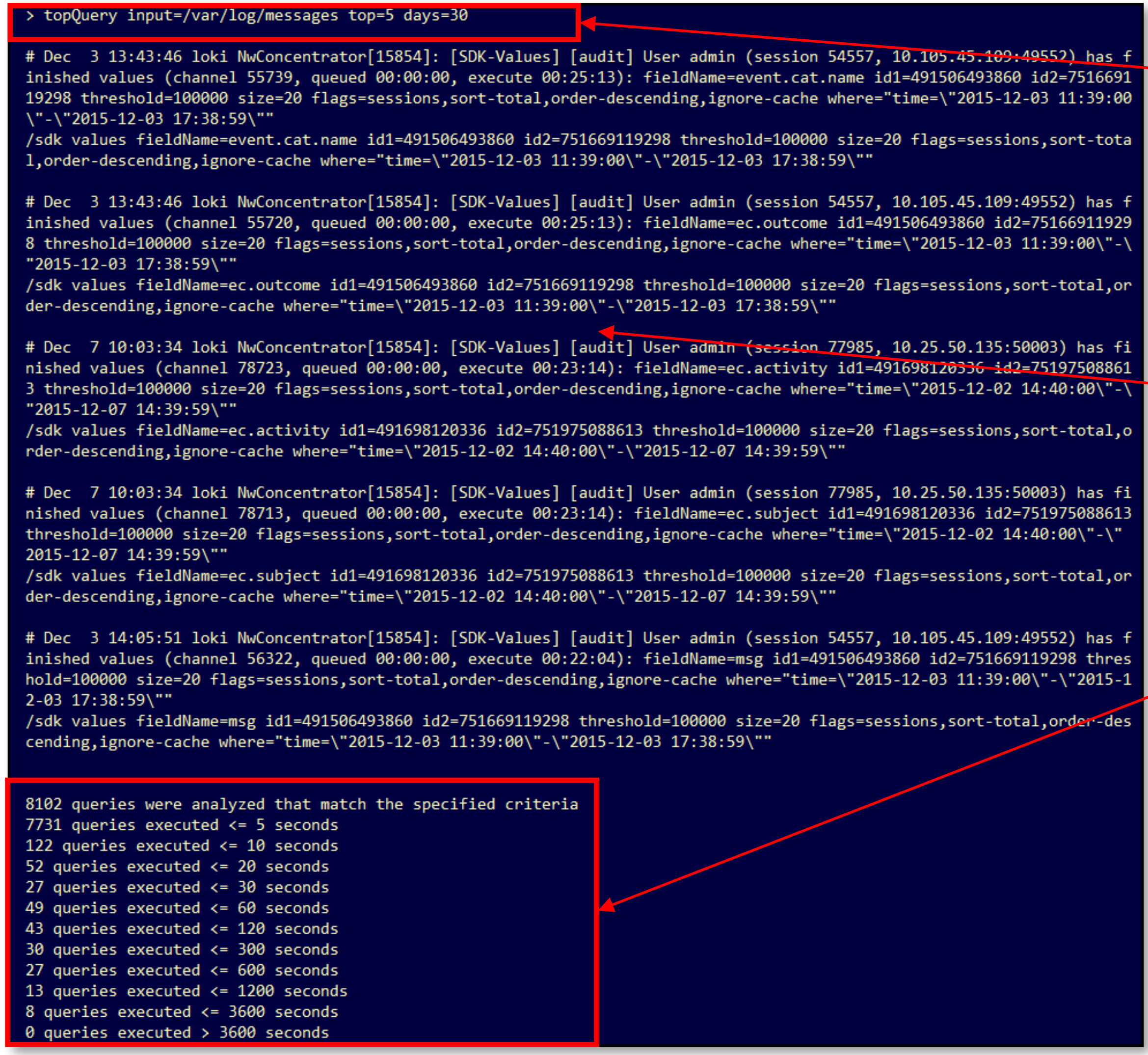
Symptoms:

1) Analysts: “We can’t use the system – it’s too slow”
2) Reports timing out (blank reports in the morning)
3) Inconsistent reporting against meta keys (gaps in data where certain values should exist)

	Checklist
	Query time statistics (**query** distribution) + analysis
	Configure app rules for common **queries**
	Check Reporting Engine Config
	Check **index** slices
	Check **index** age (vs meta age)
	Check **index** depth/configuration

RSA Charge 2016

Query (in)Sanity - topQuery



```
> topQuery input=/var/log/messages top=5 days=30

# Dec  3 13:43:46 loki NwConcentrator[15854]: [SDK-Values] [audit] User admin (session 54557, 10.105.45.109:49552) has f
inished values (channel 55739, queued 00:00:00, execute 00:25:13): fieldName=event.cat.name id1=491506493860 id2=7516691
19298 threshold=100000 size=20 flags=sessions,sort-total,order-descending,ignore-cache where="time=\"2015-12-03 11:39:00
\"-\"2015-12-03 17:38:59\""
/sdk values fieldName=event.cat.name id1=491506493860 id2=751669119298 threshold=100000 size=20 flags=sessions,sort-tota
l,order-descending,ignore-cache where="time=\"2015-12-03 11:39:00\"-\"2015-12-03 17:38:59\""

# Dec  3 13:43:46 loki NwConcentrator[15854]: [SDK-Values] [audit] User admin (session 54557, 10.105.45.109:49552) has f
inished values (channel 55720, queued 00:00:00, execute 00:25:13): fieldName=ec.outcome id1=491506493860 id2=75166911929
8 threshold=100000 size=20 flags=sessions,sort-total,order-descending,ignore-cache where="time=\"2015-12-03 11:39:00\"-\
"2015-12-03 17:38:59\""
/sdk values fieldName=ec.outcome id1=491506493860 id2=751669119298 threshold=100000 size=20 flags=sessions,sort-total,or
der-descending,ignore-cache where="time=\"2015-12-03 11:39:00\"-\"2015-12-03 17:38:59\""

# Dec  7 10:03:34 loki NwConcentrator[15854]: [SDK-Values] [audit] User admin (session 77985, 10.25.50.135:50003) has fi
nished values (channel 78723, queued 00:00:00, execute 00:23:14): fieldName=ec.activity id1=491698120336 id2=75197508861
3 threshold=100000 size=20 flags=sessions,sort-total,order-descending,ignore-cache where="time=\"2015-12-02 14:40:00\"-\
"2015-12-07 14:39:59\""
/sdk values fieldName=ec.activity id1=491698120336 id2=751975088613 threshold=100000 size=20 flags=sessions,sort-total,o
rder-descending,ignore-cache where="time=\"2015-12-02 14:40:00\"-\"2015-12-07 14:39:59\""

# Dec  7 10:03:34 loki NwConcentrator[15854]: [SDK-Values] [audit] User admin (session 77985, 10.25.50.135:50003) has fi
nished values (channel 78713, queued 00:00:00, execute 00:23:14): fieldName=ec.subject id1=491698120336 id2=751975088613
threshold=100000 size=20 flags=sessions,sort-total,order-descending,ignore-cache where="time=\"2015-12-02 14:40:00\"-\"
2015-12-07 14:39:59\""
/sdk values fieldName=ec.subject id1=491698120336 id2=751975088613 threshold=100000 size=20 flags=sessions,sort-total,or
der-descending,ignore-cache where="time=\"2015-12-02 14:40:00\"-\"2015-12-07 14:39:59\""

# Dec  3 14:05:51 loki NwConcentrator[15854]: [SDK-Values] [audit] User admin (session 54557, 10.105.45.109:49552) has f
inished values (channel 56322, queued 00:00:00, execute 00:22:04): fieldName=msg id1=491506493860 id2=751669119298 thres
hold=100000 size=20 flags=sessions,sort-total,order-descending,ignore-cache where="time=\"2015-12-03 11:39:00\"-\"2015-1
2-03 17:38:59\""
/sdk values fieldName=msg id1=491506493860 id2=751669119298 threshold=100000 size=20 flags=sessions,sort-total,order-des
cending,ignore-cache where="time=\"2015-12-03 11:39:00\"-\"2015-12-03 17:38:59\""


8102 queries were analyzed that match the specified criteria
7731 queries executed <= 5 seconds
122 queries executed <= 10 seconds
52 queries executed <= 20 seconds
27 queries executed <= 30 seconds
49 queries executed <= 60 seconds
43 queries executed <= 120 seconds
30 queries executed <= 300 seconds
27 queries executed <= 600 seconds
13 queries executed <= 1200 seconds
8 queries executed <= 3600 seconds
0 queries executed > 3600 seconds
```

- Most useful build in 10.6 (part of NwConsole – rpm can be installed standalone on any CentOS host and pointed at live NW stack)
- Run against query logs or direct live API call
- Many options to narrow the range, query type, etc.

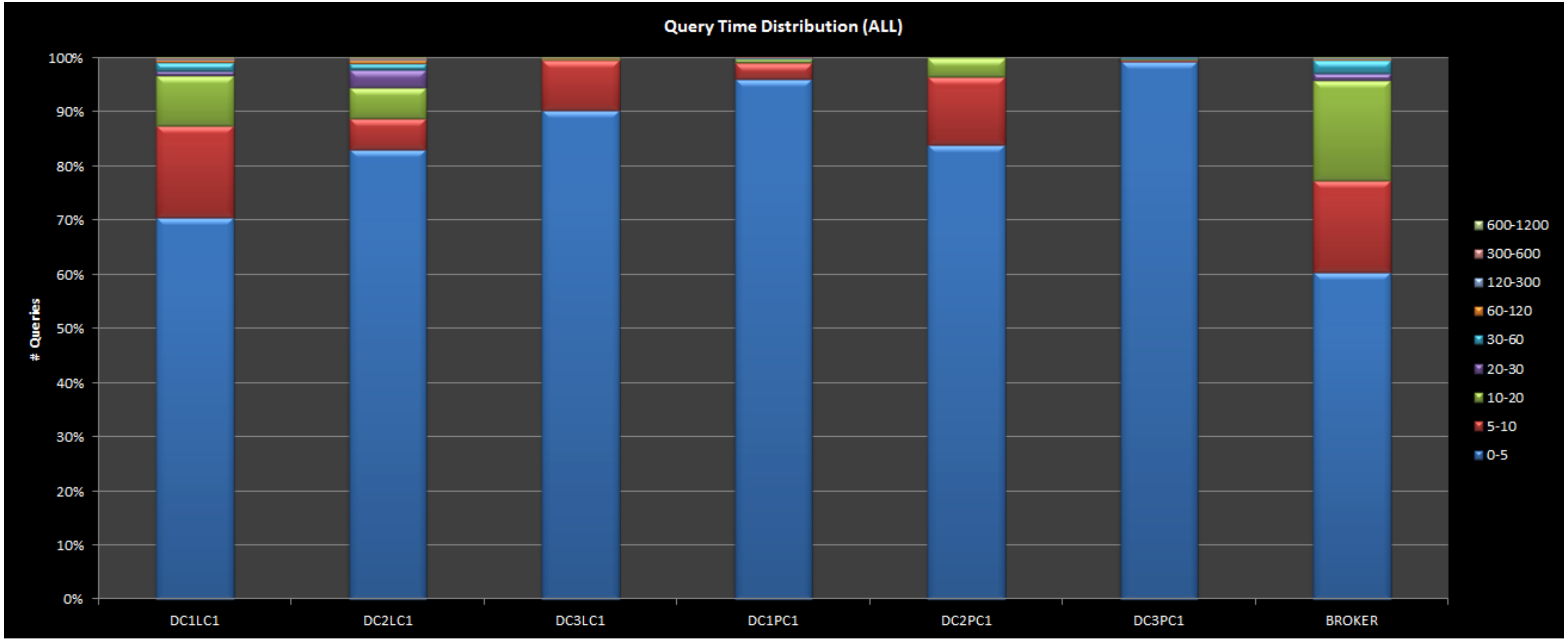
Returns the poorest performing queries based on overall execution time for both Investigation (SDK-Values) and RE (SDK-Query)

Query time distribution of result set

(CLI) > NwConsole -c login concentratorIP:50005:[ssl] admin netwitness -c topQuery days=7 top=20

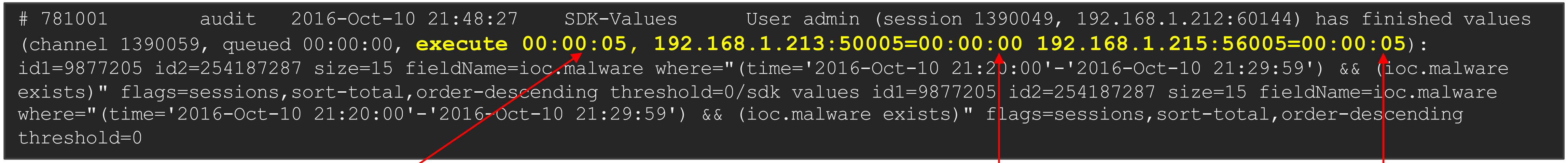
Case Study – Noname Inc.

topQuery Results



Case Study – Noname Inc.

topQuery Results



```
# 781001        audit   2016-Oct-10 21:48:27    SDK-Values      User admin (session 1390049, 192.168.1.212:60144) has finished values
(channel 1390059, queued 00:00:00, execute 00:00:05, 192.168.1.213:50005=00:00:00 192.168.1.215:56005=00:00:05):
id1=9877205 id2=254187287 size=15 fieldName=ioc.malware where="(time='2016-Oct-10 21:20:00'-'2016-Oct-10 21:29:59') && (ioc.malware
exists)" flags=sessions,sort-total,order-descending threshold=0/sdk values id1=9877205 id2=254187287 size=15 fieldName=ioc.malware
where="(time='2016-Oct-10 21:20:00'-'2016-Oct-10 21:29:59') && (ioc.malware exists)" flags=sessions,sort-total,order-descending
threshold=0
```

Broker query time – only as fast as it's slowest concentrator

Concentrator 1

Concentrator 2

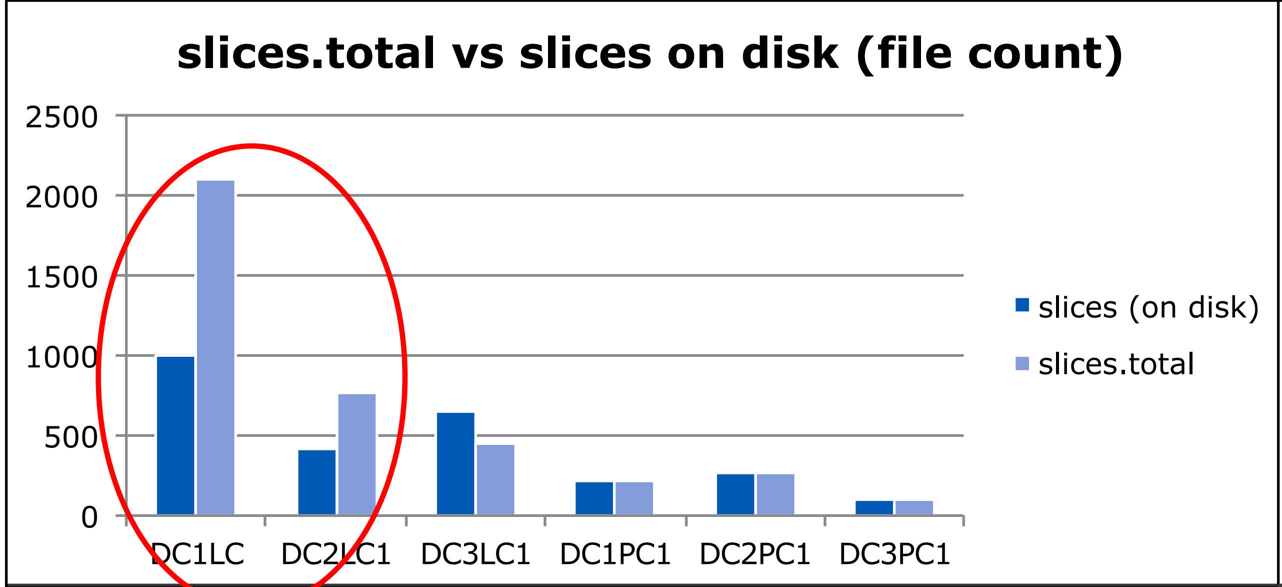
Observations (from real environment, not above):

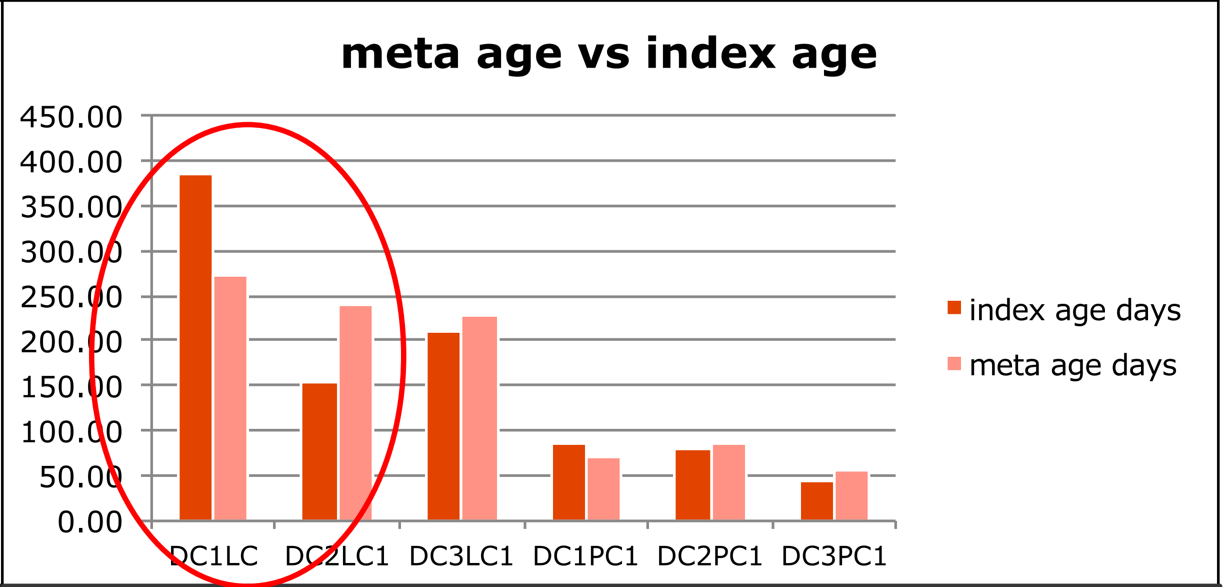
1) Terribly inefficient queries (multiple contains, regex, begins, logical statements)
2) Slowest top level queries for **log data** (most of the reports were log-based) showed 1 of 2 things:
   - The same log concentrator always responsible (DC1LC1)

   or
   - A packet concentrator was responsible

Case Study – Noname Inc.

Index Slices & Index/Meta Age





```
(API) https://concentrator:50105/index/stats/slices.total
(disk)>  find /var/netwitness/concentrator/index -mindepth 1 -type d | wc -l
```

```
(API) https://concentrator:50105/index/stats/time.begin
(API) https://concentrator:50105/database/stats/meta.oldest.file.time
```

Observations:

1) Too many slices on disk: DC1LC1, DC2LC1
2) Disparity between API reported value and slices on disk: DCLC1, DC2LC1, DC3LC1
3) Index age > Meta age on DC1LC1 (and both are much larger than business requirement)
4) Index age < Meta age on DC2LC1 = ~100 days of meta that isn't queryable
5) Packet stacks all look good.

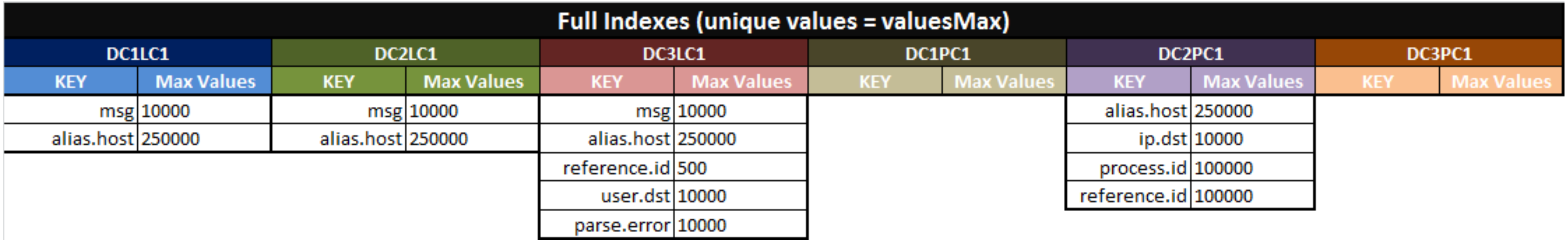
Corrective Actions:

1) CRON job to timeRoll MetaDB (10.5 should also roll index) – consistent across all devices
2) Clean-up/Delete Old Index slices (delete from disk)
3) Remove scheduled task for time-based slicing, use the session-count config.
4) Engage Customer Support (re-index might be needed here)

RSA® Charge 2016

Case Study – Noname Inc.

Index Depth/Configuration



Full Indexes (unique values = valuesMax)

DC1LC1		DC2LC1		DC3LC1		DC1PC1		DC2PC1		DC3PC1	
KEY	Max Values	KEY	Max Values	KEY	Max Values	KEY	Max Values	KEY	Max Values	KEY	Max Values
msg	10000	msg	10000	msg	10000			alias.host	250000		
alias.host	250000	alias.host	250000	alias.host	250000			ip.dst	10000		
				reference.id	500			process.id	100000		
				user.dst	10000			reference.id	100000		
				parse.error	10000						

Observations:

1) Lines up with the “Data is missing” complaint. Low alias.host max values, ip.dst randomly restricted to 10,000 on DC2PC1
2) Note (not shown) – DC3LC1 had a HUGE index defined. Many unnecessary IndexValues and large ValuesMax = Too much data in the index, space filled up before metaDB did.

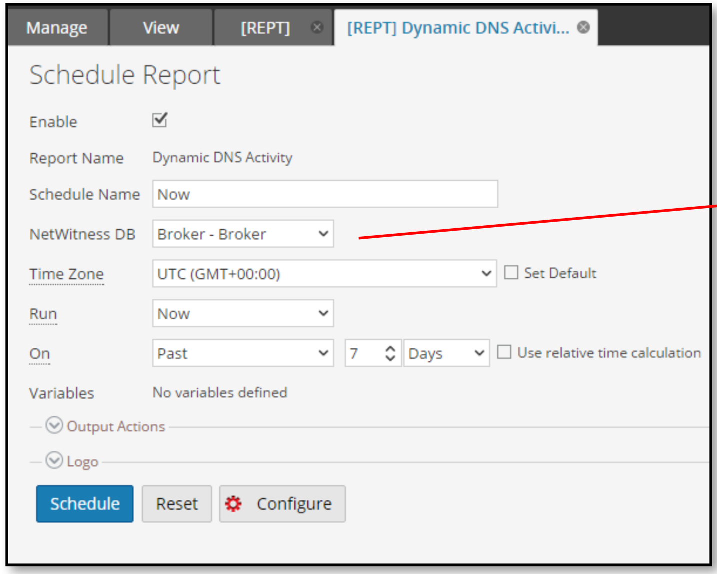
   ** This was done due to misunderstanding of the reporting engine. Only meta in the “Where” clause must be indexed, not the “Select” clause.

Corrective Actions:

1) Full index review (remove unnecessary indexes, remove completely unique indexes like ‘msg’, increase valuesMax for alias.host
2) Make sure indexes are consistent across like-decoders

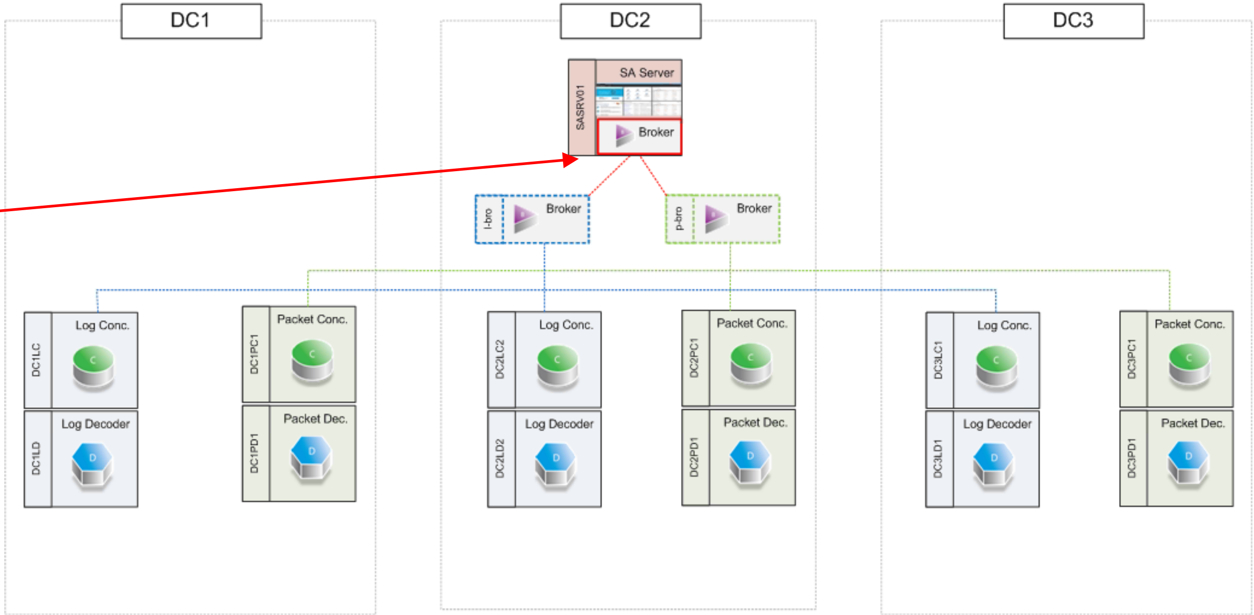
Case Study – Noname Inc.

Reporting Engine Configuration – Careful where you point that thing.



Schedule Report

Enable	☑
Report Name	Dynamic DNS Activity
Schedule Name	Now
NetWitness DB	Broker - Broker
Time Zone	UTC (GMT+00:00) ☐ Set Default
Run	Now
On	Past 7 Days ☐ Use relative time calculation
Variables	No variables defined



Observations:

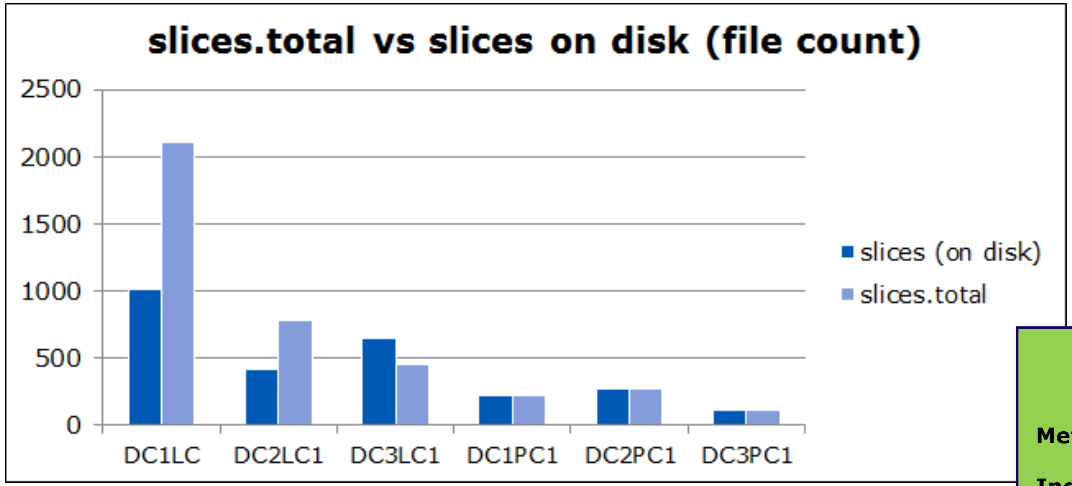
1) Every single report, whether log or packet, was pointed at the Primary Broker
2) Log reports were timing out mainly due to packet concentrators taking a long time to respond to the query !!
3) Many, many inefficient queries, using lists when feeds would be better, etc.

Corrective Actions:

1) Go through each report, point log reports at log devices, packet reports at packet devices
2) Fixed overlapping report ranges (eg. weekly reports asking for 30 days of data)
3) Moved as much logic to app rules as possible, moved most (but not all) lists to feeds

Case Study – Noname Inc.

After things got happy again.

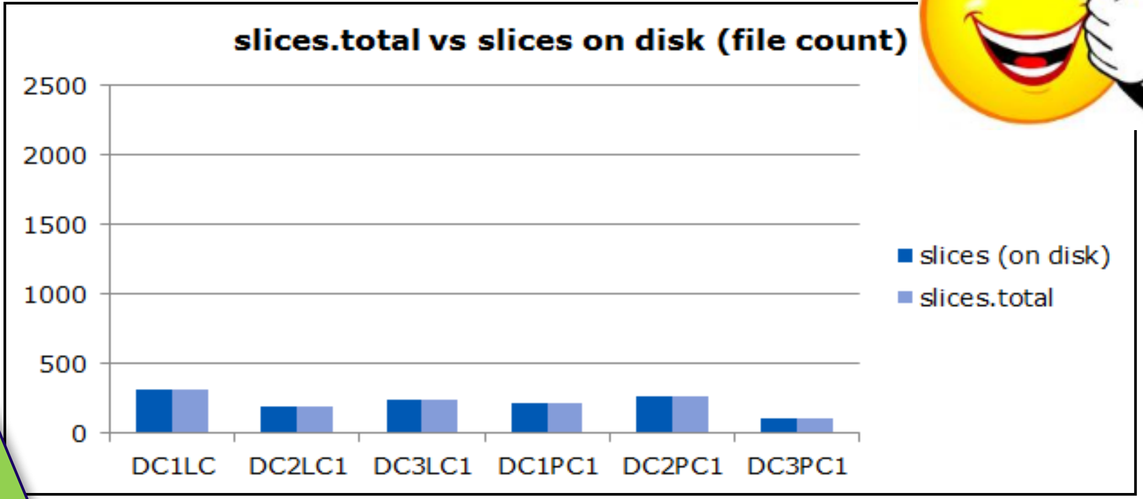


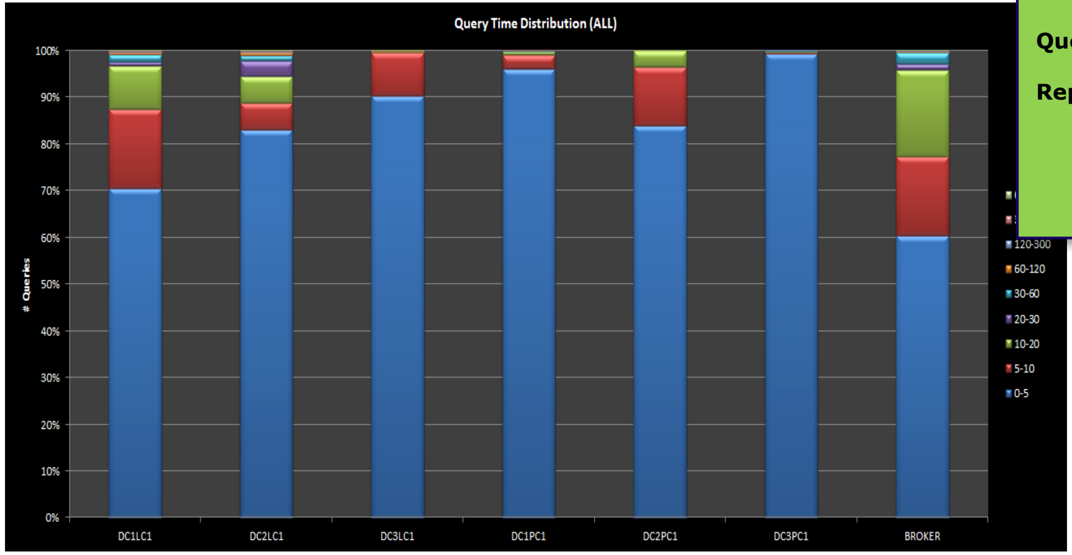
Meta timeroll

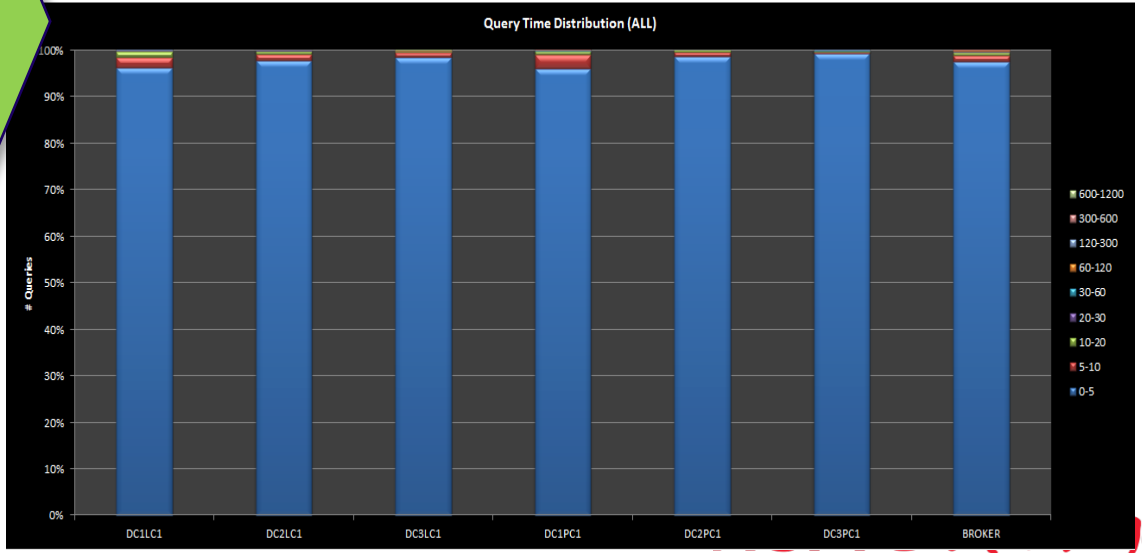
Index slice -> session cnt

Queries -> app rules

Reports -> split log/packet







Please Complete Session Evaluation

#RSACharge

RSA® Charge 2016

#RSACharge