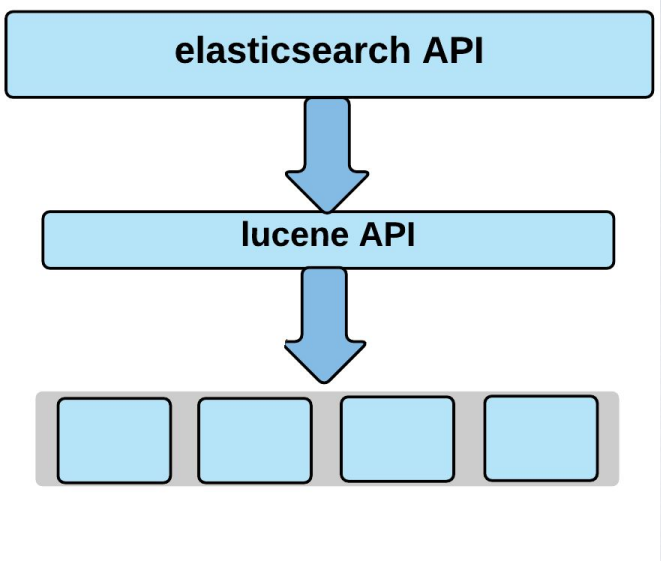
ELASTICSEARCH

Tech Talk Fridays

LUCENE

- Elasticsearch is a distributed web server written on Lucene. It's accessed via HTTP requests.
- Lucene is a java library. It can be included in projects. It can be used via function calls.



ALGORITHMS IN LUCENE

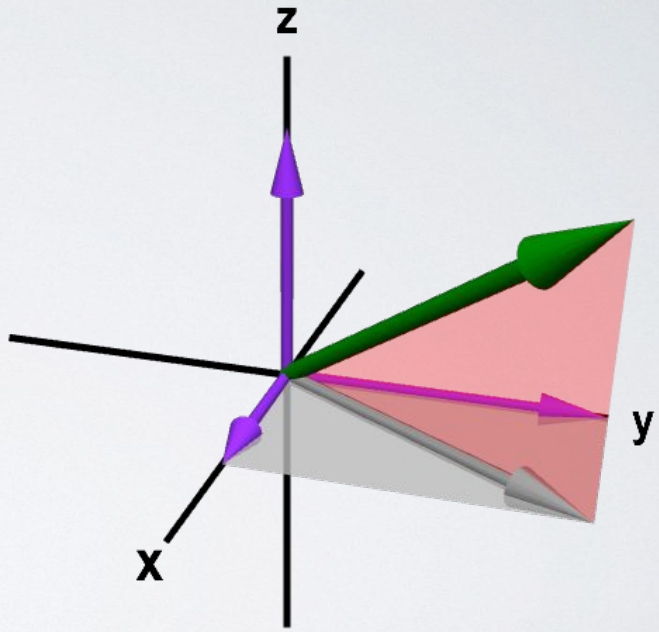
- TFIDFSimilarity : TF-IDF-Similarity
- Levenshtein Algorithm

TF-IDF-SIMILARITY (1/6)

- Lucene does a one-zero, true/false, boolean filtering of documents.
- Filtered documents are scored in using TF-IDF values as weights.

TF-IDF-SIMILARITY (2/6)

- This scoring of documents is called Vector Space Model, because each index term is a vector and TF-IDF value is the weight.



TF-IDF-SIMILARITY (3/6)

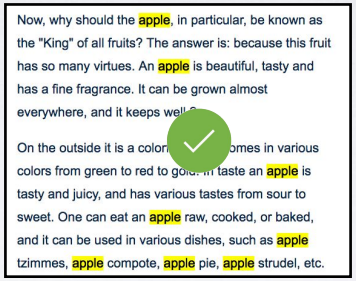
- TF varies with the number of occurrences of a term in the given document.

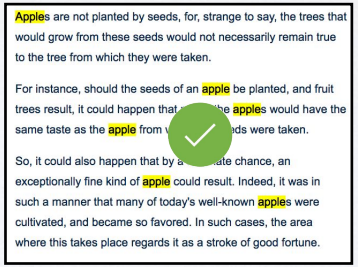
Now, why should the apple, in particular, be known as the "King" of all fruits? The answer is: because this fruit has so many virtues. An apple is beautiful, tasty and has a fine fragrance. It can be grown almost everywhere, and it keeps well.?

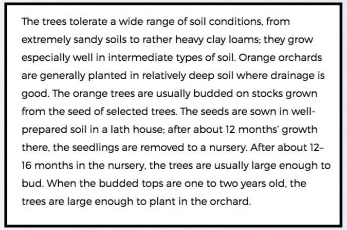
On the outside it is a colorful fruit. It comes in various colors from green to red to gold. In taste an apple is tasty and juicy, and has various tastes from sour to sweet. One can eat an apple raw, cooked, or baked, and it can be used in various dishes, such as apple tzimmes, apple compote, apple pie, apple strudel, etc.

TF-IDF-SIMILARITY (4/6)

- IDF varies with the number of documents containing the given term.







TF-IDF-SIMILARITY (5/6)

- **norm :** An importance value of each field in the document is computed at indexing time and saved.
- Length of field.
  Boost of field.
- Shorter fields contribute more score.

Now, why should the apple, in particular, be known as the "King" of all fruits? The answer is: because this fruit has so many virtues. An apple is beautiful, tasty and has a fine fragrance. It can be grown almost everywhere, and it keeps well.?

On the outside it is a colorful fruit. It comes in various colors from green to red to gold. In taste an apple is tasty and juicy, and has various tastes from sour to sweet. One can eat an apple raw, cooked, or baked, and it can be used in various dishes, such as apple tzimmes, apple compote, apple pie, apple strudel, etc.

Length of field = 583 characters.
Field Boost = 1 (default, but can be changed)

TF-IDF-SIMILARITY (6/6)

Lucene's Scoring Function:

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost}() \cdot \text{norm}(t,d) \right)$$

Lucene's Scoring Function:

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost}() \cdot \text{norm}(t,d) \right)$$

t.getBoost - this is the search time boost given to individual terms of the query.

apple (boost 2)
orange (boost 1)

```
GET /_search
{
  "query": {
    "bool": {
      "should": [
        {
          "term": {
            "field_name": {
              "value": "apple",
              "boost": 2.0 ❶
            }
          }
        },
        {
          "term": {
            "field_name": "orange" ❷
          }
        }
      ]
    }
  }
}
```

Lucene's Scoring Function:

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost}() \cdot \text{norm}(t,d) \right)$$

queryNorm - is a normalising factor to make scores between queries comparable.

The factor does not affect document ranking.

Only tries to make scores from different queries (and even different indices) comparable.

Lucene's Scoring Function:

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost}() \cdot \text{norm}(t,d) \right)$$

coord - This awards additional score to documents for matching more than one query term.

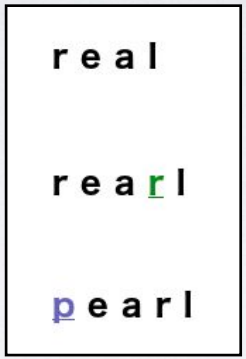
ID	Query Term 1 ("Apple")	Query Term 2 ("Orange")	Hypothetical coord score
1	Yes	Yes	2
2	Yes	x	1
3	x	Yes	1
4	x	x	0

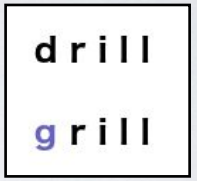
ALGORITHMS IN LUCENE

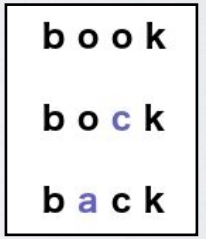
- TFIDFSimilarity : TF-IDF-Similarity
- **Levenshtein Algorithm**

LEVENSHTEIN DISTANCE

- Given two words, find the minimum number of operations required to change the first word into second word.
- Operations allowed:
  Insert,
  Delete,
  Substitute







LEVENSHTEIN DISTANCE

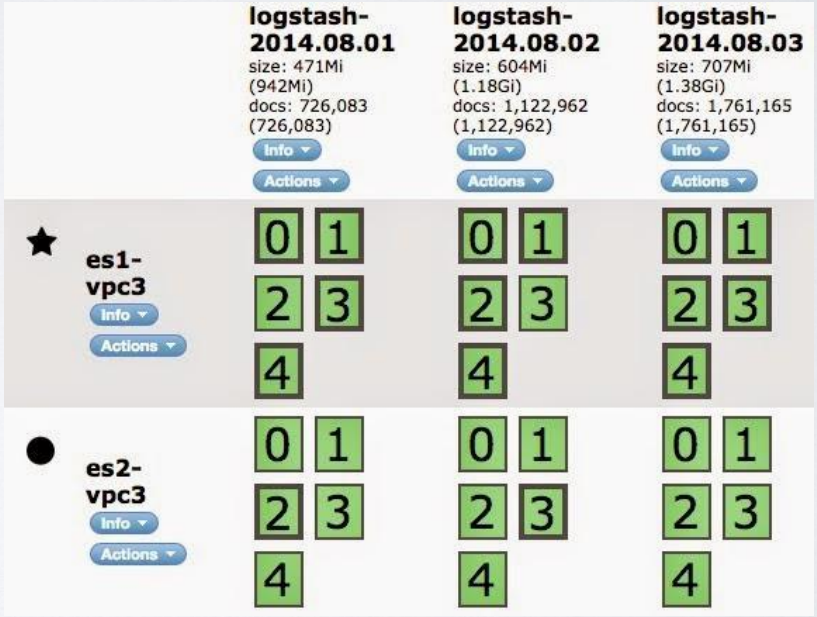
	B	O	O	K
B	0	1	2	3
A	1	1	2	3
C	2	2	2	3
K	3	3	3	2

LEVENSHTEIN DISTANCE

```
function LevenshteinDistance(char s[1..m], char t[1..n]):
  // for all i and j, d[i,j] will hold the Levenshtein distance between
  // the first i characters of s and the first j characters of t
  // note that d has (m+1)*(n+1) values
  declare int d[0..m, 0..n]

  set each element in d to zero

  // source prefixes can be transformed into empty string by
  // dropping all characters
  for i from 1 to m:
      d[i, 0] := i

  // target prefixes can be reached from empty source prefix
  // by inserting every character
  for j from 1 to n:
      d[0, j] := j

  for j from 1 to n:
      for i from 1 to m:
          if s[i] = t[j]:
            substitutionCost := 0
          else:
            substitutionCost := 1
          d[i, j] := minimum(d[i-1, j] + 1,                   // deletion
                             d[i, j-1] + 1,                   // insertion
                             d[i-1, j-1] + substitutionCost)  // substitution

  return d[m, n]
```

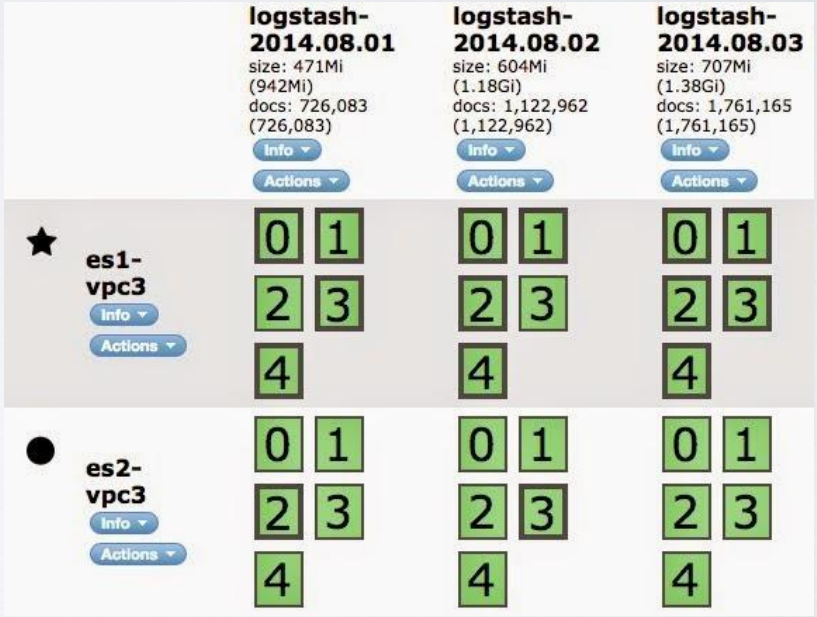
SOME ELASTICSEARCH TERMS AND IT'S ARCHITECTURE (1/3)

- An index in elastic search is similar to a DB in mysql.
- Index is divided into different shards.



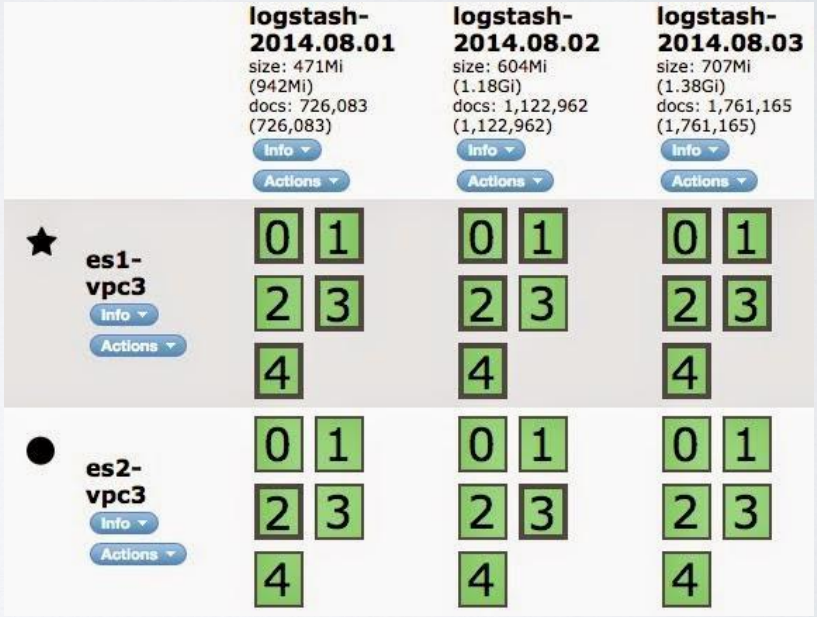
SOME ELASTICSEARCH TERMS AND IT'S ARCHITECTURE (2/3)

- A "type" in elastic search is similar to a table in mysql.
- An Index can have multiple types.



SOME ELASTICSEARCH TERMS AND IT'S ARCHITECTURE (3/3)

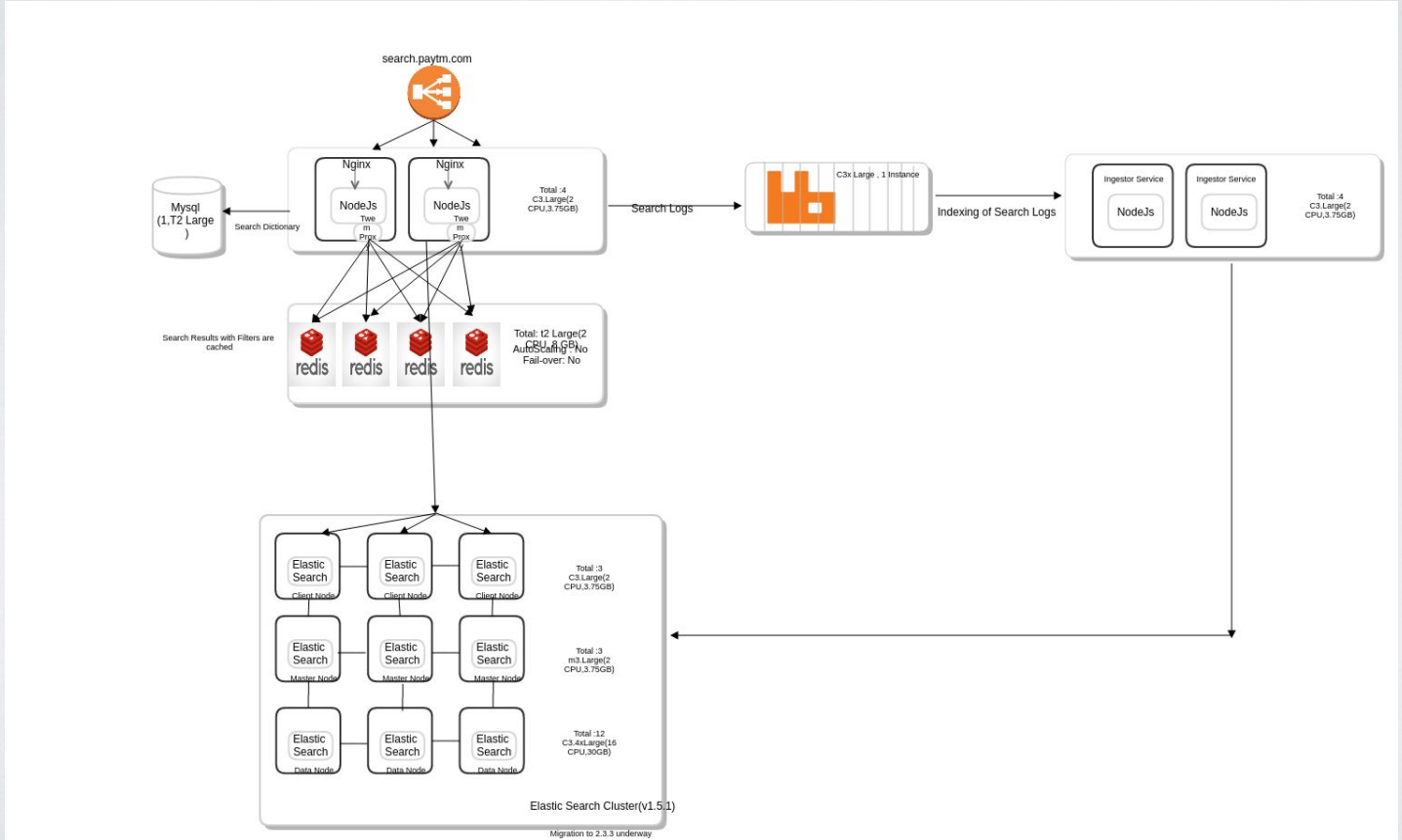
- A shard of an index contains partial data of the index that no other shard has.
- There are primary shards and replica shards. Replica and primary have exactly the same data.
- All shards are balanced across all available nodes.



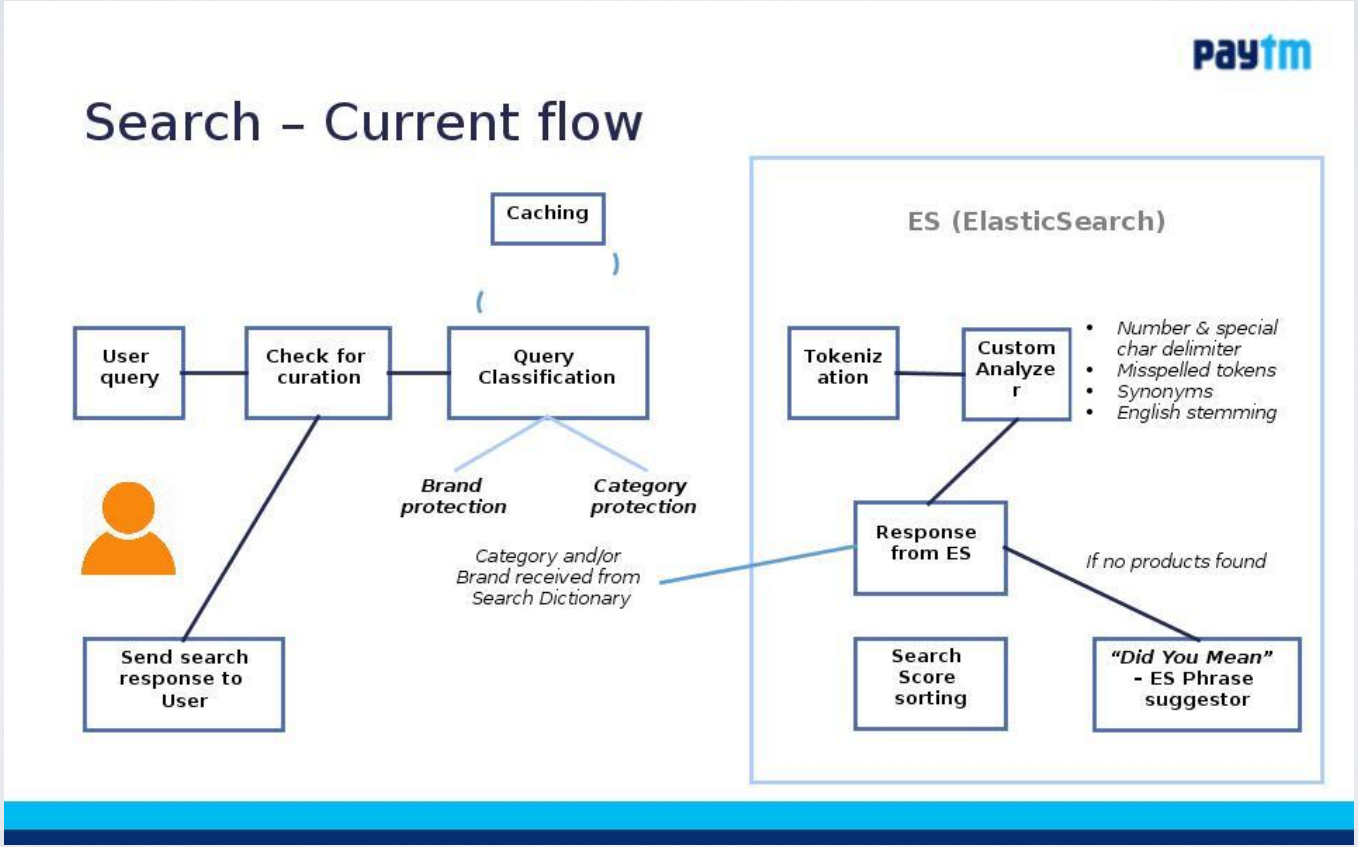
ELASTICSEARCH

- We are using elastic search to power Grid, Search, Autosuggestions and Seller Panel.
- It provides very fast lookup of data.
- Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. It centrally stores your data so you can discover the expected and uncover the unexpected.

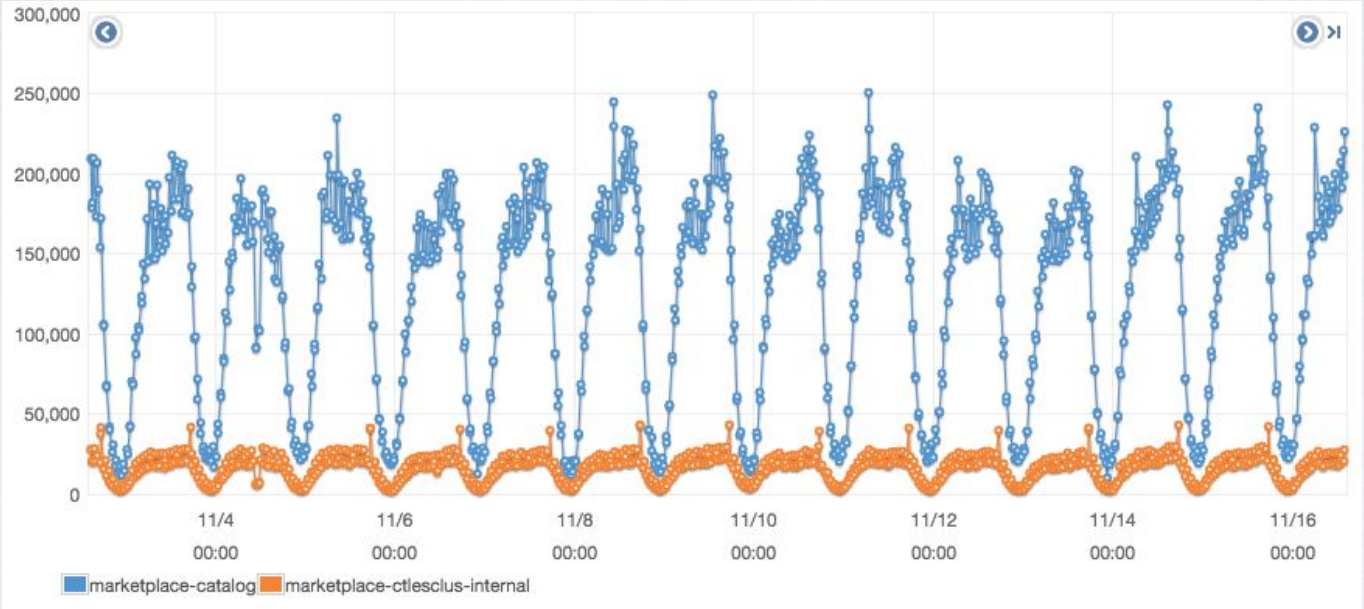
SEARCH ARCHITECTURE



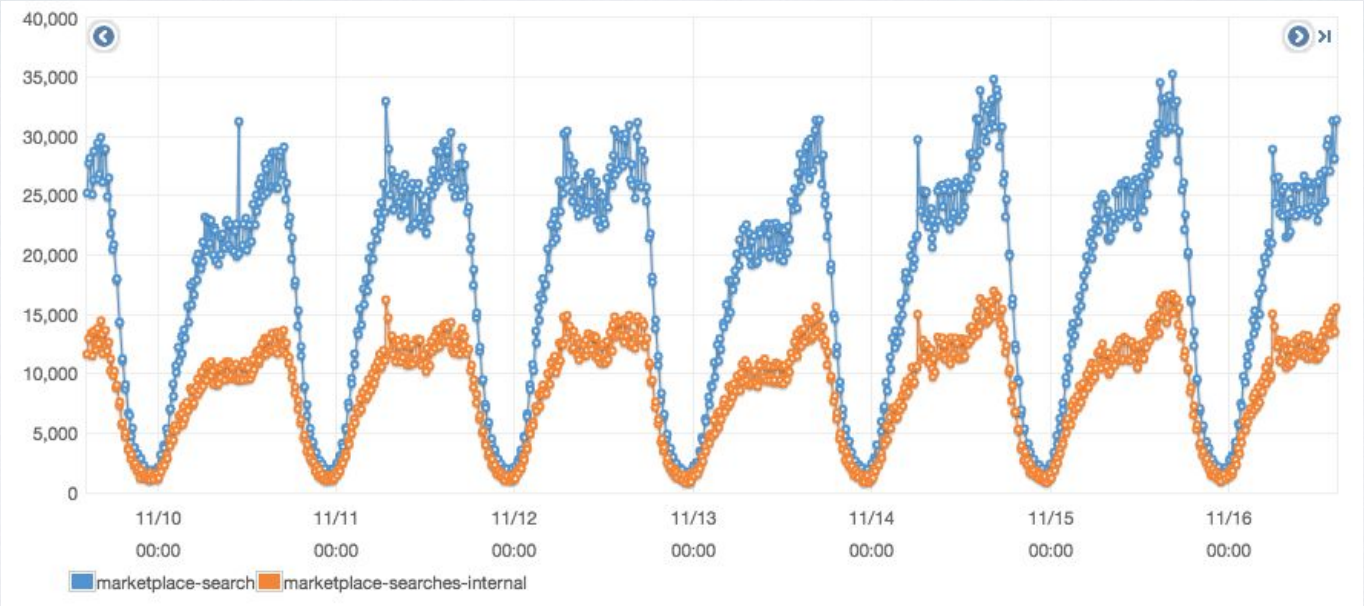
SEARCH FLOW


Search – Current flow
Caching
User query
Check for curation
Query Classification
Brand protection
Category protection
Category and/or Brand received from Search Dictionary
Send search response to User
ES (ElasticSearch)
Tokeniz ation
Custom Analyze r
Number & special char delimiter
Misspelled tokens
Synonyms
English stemming
Response from ES
If no products found
Search Score sorting
"Did You Mean" - ES Phrase suggestor

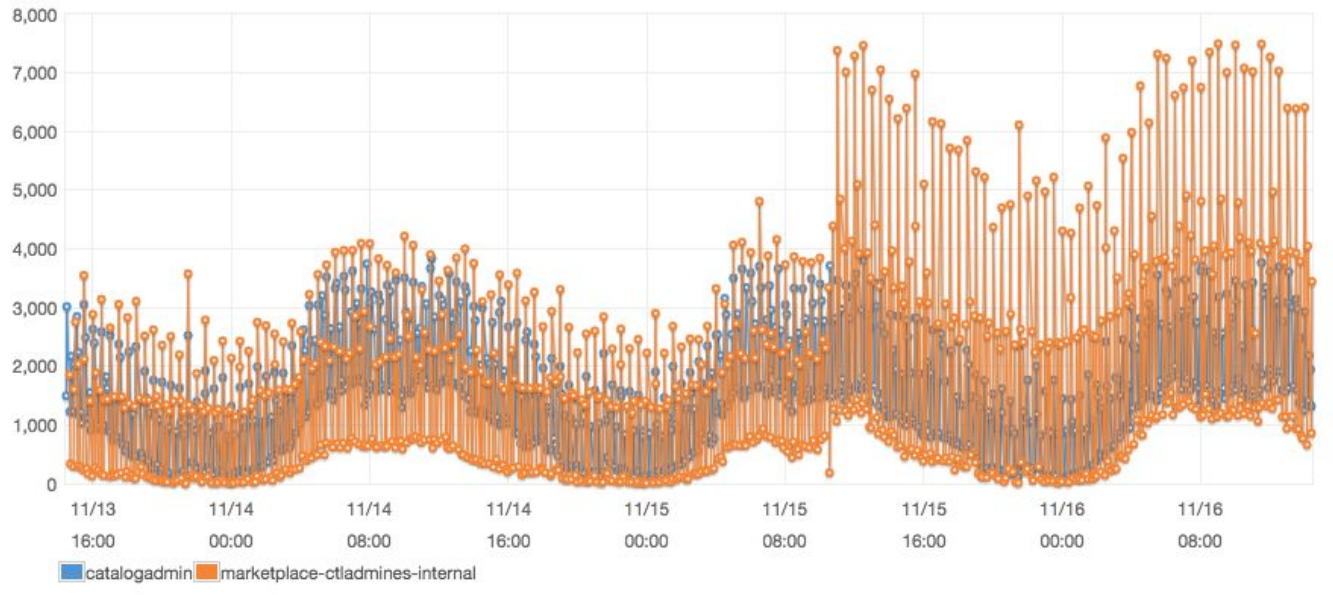
HOW MANY REQUEST ARE WE SERVING WITH ES?



HOW MANY REQUEST ARE WE SERVING WITH ES?



HOW MANY REQUEST ARE WE SERVING WITH ES?



DIFFERENCE BETWEEN SEARCH-ES AND CATALOG-ES

- Search ES contains out of stock products.
- Searches globally on the paytm database of products for matching keyword.
- Documents are scored based on brand, category, description and product name. Different weights for each.
- Catalog ES doesn't have out of stock products.
- Only serves products of a specific category ID and it's children.
- Products are displayed in a predefined order stored in ES. This order can be changed anytime using Racking flow.

DEMO ON PAYTM WEBSITE

We'll do a demo on which requests go to Search ES and which requests go to Catalog ES.

CLUSTER SETUP OF ELASTICSEARCH

- Elastic search uses zen discovery to find other nodes present in the cluster.
- We tell each node the IP of other nodes present in the cluster in a config file.
- Even if you run a single instance of elasticsearch on your local machine, you'll be running a cluster with one node.
- If any node goes down, cluster health is affected. It would turn from green to yellow or red.
- Yellow cluster state - replica shards of an index are down.
- Red cluster state - Primary shards of an index are down.

DEFINITIONS OF DIFFERENT TYPES OF NODE IN ELASTICSEARCH (1/3)

Master Node

- Responsible for creating or deleting an index.
- Tracking which nodes are part of the cluster.
- Decide which shards to allocate to which nodes.

Tip - Ensure there are dedicated master nodes to ensure stable cluster health.

DEFINITIONS OF DIFFERENT TYPES OF NODE IN ELASTICSEARCH (2/3)

Data Node

- Data nodes hold the shards.
- Handles CRUD, search and aggregations.

Tip -These operations put load on memory, IO and CPU. Monitoring them is a good idea. More data nodes should be added if these are consistently high.

DEFINITIONS OF DIFFERENT TYPES OF NODE IN ELASTICSEARCH (3/3)

Client Node

- Routing of CRUD queries.
- Basically behave as load balancers.

Tip - A large cluster can benefit from client nodes by offloading master eligible nodes and data nodes. But adding too many client nodes can slow down a cluster.

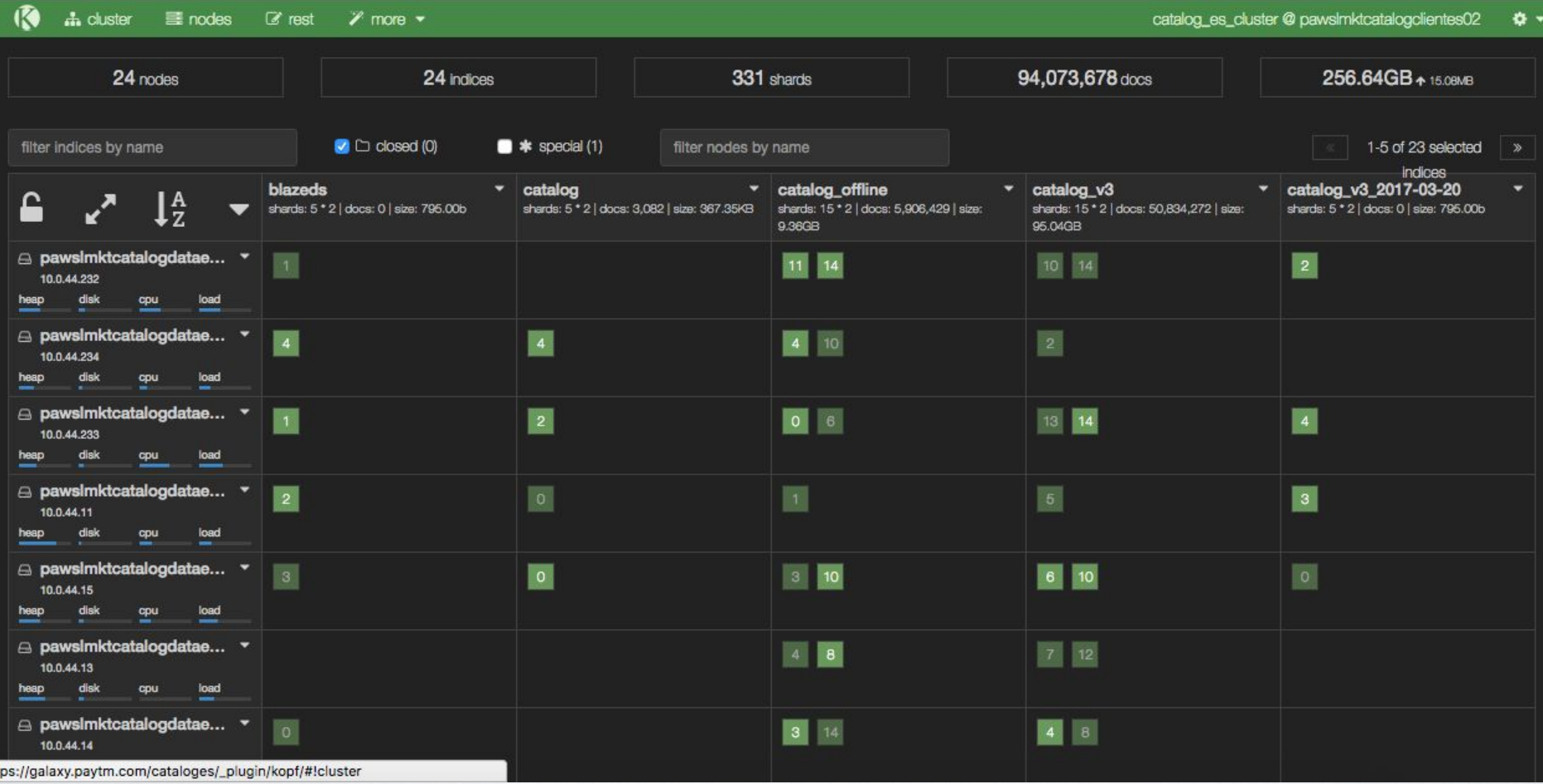
INFORMATION ON OUR CURRENT MASTER, CLIENT AND DATA NODES

ES Cluster	Master Nodes	Data Nodes	Client Nodes
Admin ES	3	4	
Catalog ES	6	16	2
Search ES	3	10	

INFORMATION ON OUR DATA SIZE, NUMBER OF DOCS AND AVG. QUERY TIME

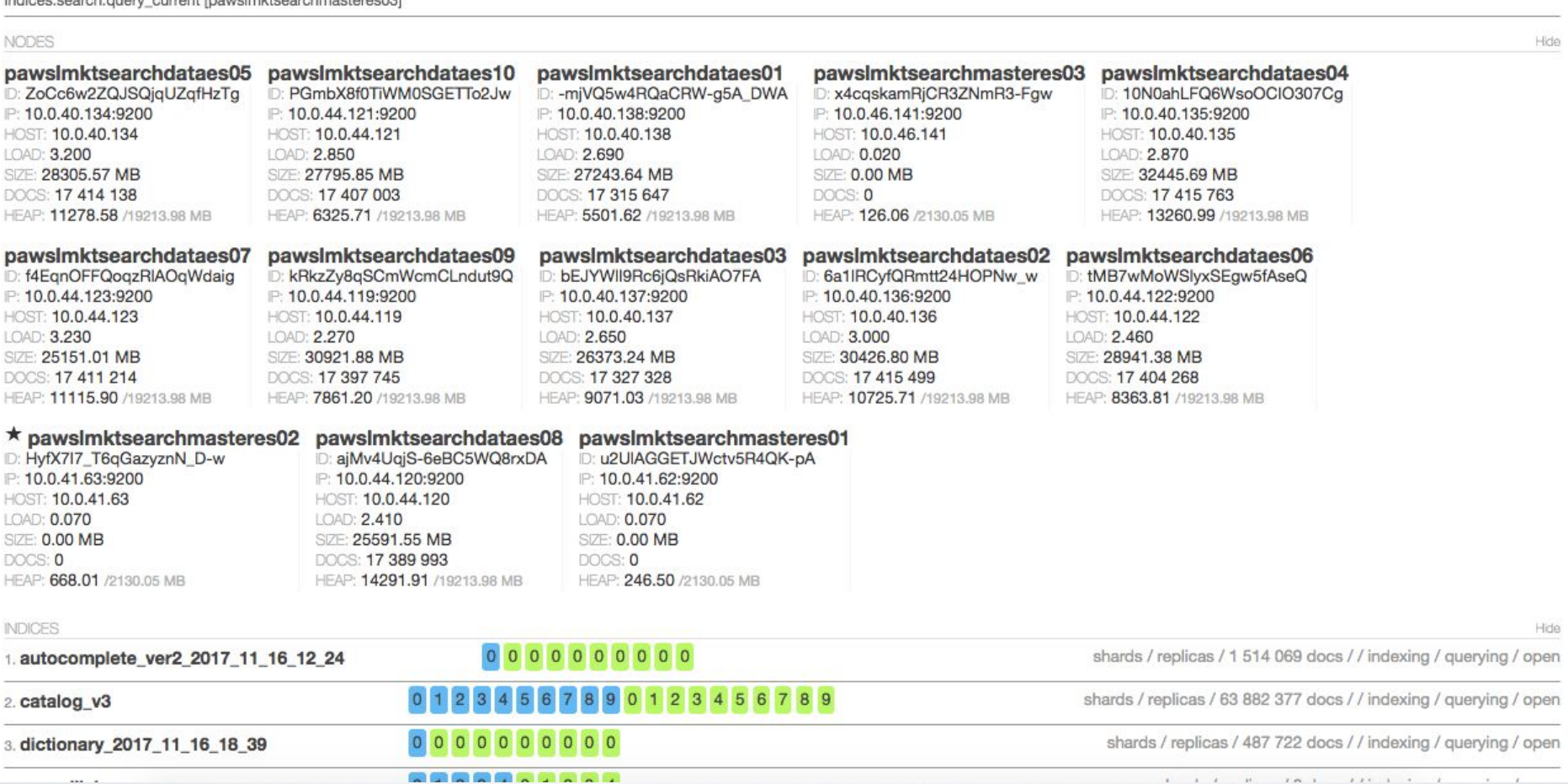
ES Cluster	Data Size	Num Docs	Avg. Query Time
Admin ES	254.4 GB	14,59,55,294	35 ms
Catalog ES	127.83 GB	4,10,19,363	18 - 38 ms
Search ES	103 GB	2,81,60,812	17 - 22 ms

MONITORING ELASTICSEARCH


cluster
nodes
rest
more
catalog_es_cluster @ pawslmktcatalogclientes02
24 nodes
24 indices
331 shards
94,073,678 docs
256.64GB 15.08MB
filter indices by name
closed (0)
special (1)
filter nodes by name
1-5 of 23 selected
indices
blazeds
shards: 5 * 2 | docs: 0 | size: 795.00b
catalog
shards: 5 * 2 | docs: 3,082 | size: 367.35KB
catalog_offline
shards: 15 * 2 | docs: 5,906,429 | size: 9.36GB
catalog_v3
shards: 15 * 2 | docs: 50,834,272 | size: 95.04GB
catalog_v3_2017-03-20
shards: 5 * 2 | docs: 0 | size: 795.00b
pawslmktcatalogdatae...
10.0.44.232
10.0.44.234
10.0.44.233
10.0.44.11
10.0.44.15
10.0.44.13
10.0.44.14
heap
disk
cpu
load
https://galaxy.paytm.com/cataloges/_plugin/kopf/#!cluster

KOPF PLUGIN

PARAMEDIC PLUGIN



PARAMEDIC PLUGIN

CLUSTER HEALTH

- GET /_cat/health?v
- Green = everything is good
- Yellow = all data is available but some replicas are missing.
- Red = Some data is not available.

ADD/REMOVE NODES

1. Fix the unicast discovery list to contain the new node.
2. Start the new node
3. Watch it join the cluster. If there's a problem, troubleshoot and repeat.
4. Use the allocation filtering API to ban all shards from the node you will be removing.
5. Wait until there are no shards on that node.
6. Shut down the node.