

Percolator

Martijn van Groningen
@mvgroningen

Topics

- What is percolator?
- Use cases
- How does the percolator work?
- Percolator features

Add a document

http verb index type id

```
curl -XPUT 'localhost:9200/my-index/my-type/1' -d '{
  "title" : "Coffee percolator",
  "body" : "A coffee percolator is a type of ..."
}'
```

and then search

```
curl -XGET 'localhost:9200/my_index/_search' -d '{
  "query" : {
    "match" : { ← type of query
      "body" : "coffee"
    }
  }
}'
```

Annotations for the curl command:

- http verb: -XGET
- index: my_index
- search endpoint: _search
- type of query: match
- field: body
- query string: coffee

and then search

```
{
  ...
  "hits" : {
    "total" : 3,
    "max_score" : 0.4,
    "hits" : [
      {
        "_index" : "my_index"
        "_type" : "my_type"
        "_id" : "1",
        "_score" : 0.4
        "_source" : {
          "title" : "Coffee percolator",
          "body" : "A coffee percolator is a..."
        }
      },
      ...
    ]
  }
}
```

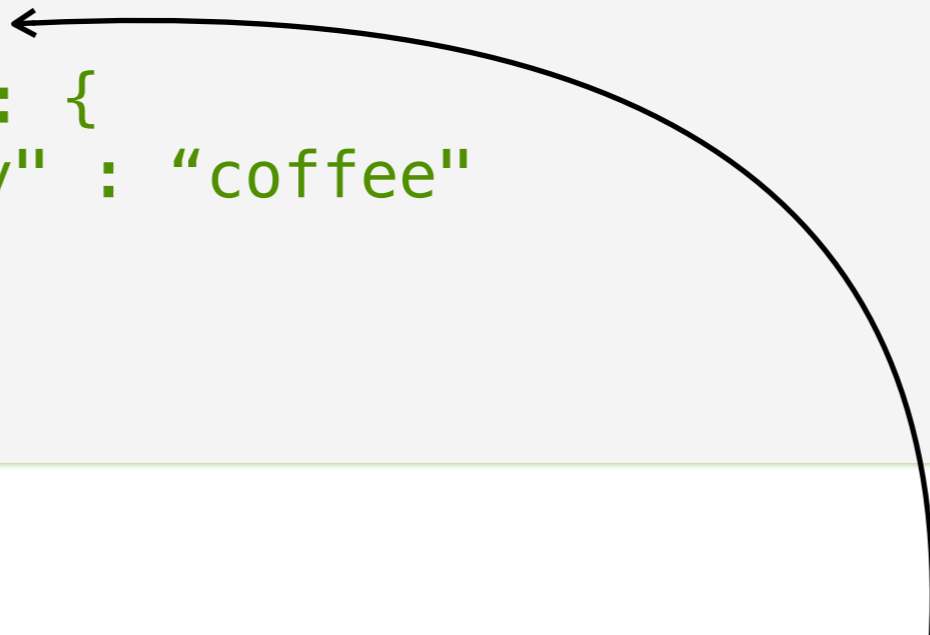
Now lets percolate!

Register a query

Reserved percolator type



```
curl -XPUT 'localhost:9200/my-index/.percolator/my-id' -d '{
  "query" : {
    "match" : {
      "body" : "coffee"
    }
  }
}'
```



Reserved top level json object

Percolate a document

index containing
queries

type of document

endpoint

```
curl -XGET 'localhost:9200/my-index/my-type/_percolate' -d '{
  "doc" : {
    "title" : "Coffee percolator",
    "body" : "A coffee percolator is a type of ..."
  }
}'
```

The document being percolated

Percolate a document

```
{
  "took" : 19,
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "count" : 4,
  "matches" : [
    {
      "_index" : "my-index",
      "_id" : "my-id"
    },
    ...
  ]
}
```

total amount of matching queries

A matching query

Percolator

- Reversed search.

Storing queries instead of data.

Querying with data instead of queries.

- This works because queries and documents are defined as json documents.

Percolation in the wild

Alerting use case

- Store and register queries that monitor data.
End users can define their alerts via application.
- Execute the percolate api right after indexing.
No need to wait - percolator works in realtime.
- Examples:
Price monitor, News alerts, Stock alerts, Weather alerts

Alerting use case

Registering a query via index api:

```
curl -XPUT 'localhost:9200/alerts/.percolator/user-1' -d '{
  "query" : {
    "bool" : {
      must : [
        {
          "range" : {
            "product.price" : { ← Price restriction
              "lte" : 500
            }
          }
        },
        {
          "match" : {
            "product.name" : "my led tv" ← name restriction
          }
        }
      ]
    }
  }
}
```

Percolator - alerting use case

Index api:

```
curl -XPUT 'localhost:9200/prices/price/1' -d '{
  "product" : {
    "name" : "my led tv",
    "price" : 499
  }
}'
```

We stored a new product,
now what?

Percolator - alerting use case

Percolate api:

```
curl -XPOST 'localhost:9200/alerts/price/_percolate' -d '{
  "doc" : {
    "product" : {
      "name" : "my led tv",
      "price" : 499
    }
  }
}'
```


ok, sending the data twice...

Percolator - alerting use case

Index response:

```
{  
  "ok" : true,  
  "_index" : "prices"  
  "type" : price  
  "_id" : 1  
  "version" : 1  
}
```

Percolator - alerting use case

- Percolate existing document api

Combination of the get and percolate api

index containing the new product

product id

```
curl -XGET 'localhost:9200/prices/price/1/_percolate?  
percolate_index=alerts'
```

index containing the queries

Query feedback use case

- Store all users' queries of a specific time frame
Last week's, last month's queries.
- Provide feedback to advertisement owner.
Execute percolate api while editing the ad.
- Examples:
Market places, web advertisements.

Classification use case

- Store queries that can identify patterns in your documents.
- Percolate a document before indexing it.
Enrich the document with the queries it matches with.
- Examples:
Automatically tag documents, geo tag documents and ways to automatically categorize documents.

Diving into the Percolator

Percolator - how does it work?

- Each shard holds a collection of parsed queries in memory.
- The queries are also stored on the shard (Lucene index)
- The collection of queries get updated by every index, create, update or delete operation in realtime.

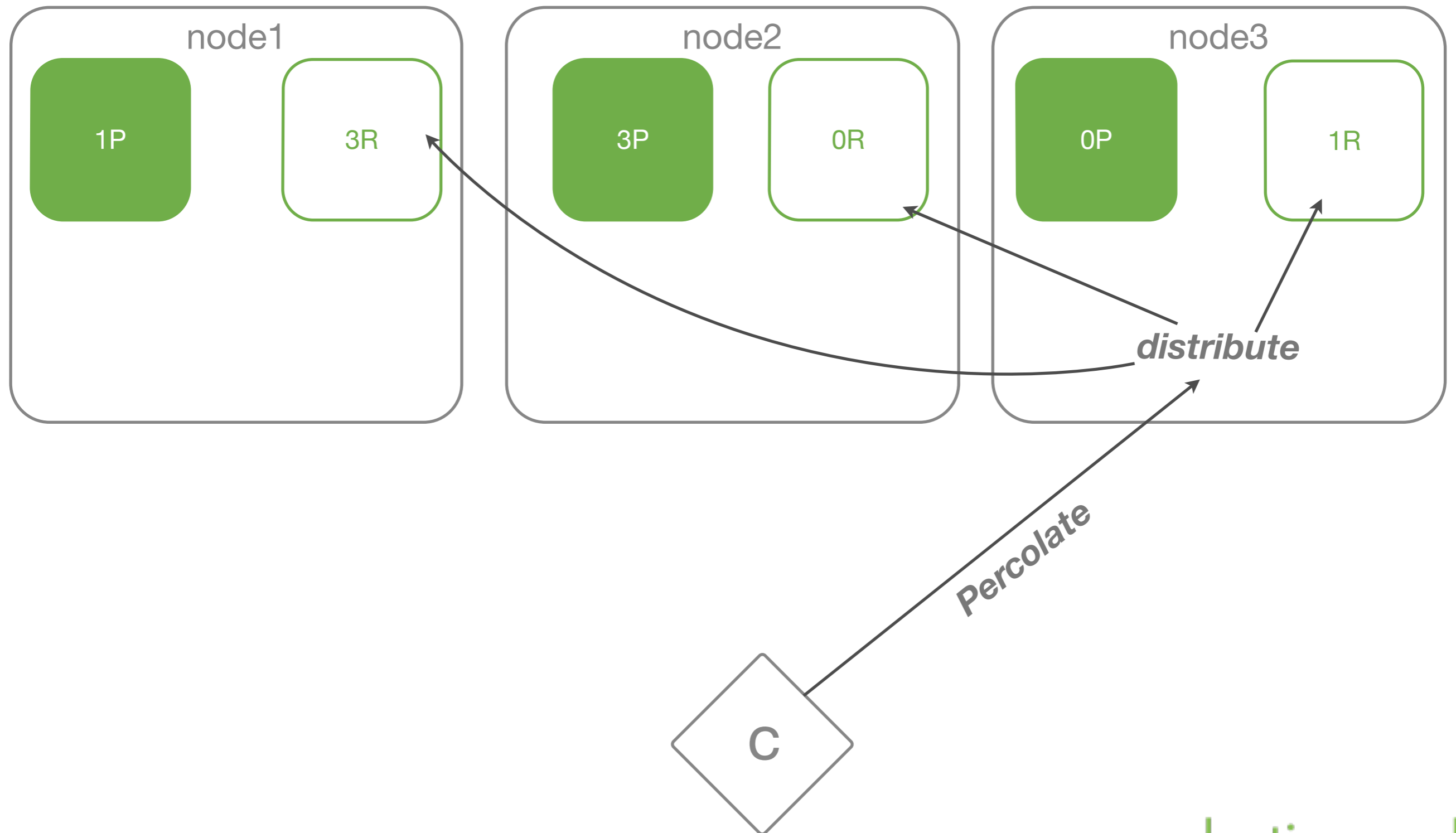
Percolator - how does it work?

- During percolating the document to be percolated gets indexed into an in memory index.
- All shard queries are executed against this one document in memory index.
Shard level execution time is linear to the amount queries to evaluate.
- After all queries have been evaluated the in memory index gets cleaned up.

Distributed percolator

- Percolate api executes the request in parallel on all shards.
- Use routing and multi tenancy to reduce the amount of queries to evaluate.
 - Routing will reduce the amount of shards.
 - More indices (and therefore more shards) reduces the amount of queries per shard.

Distributed execution



Percolator

Multi tenancy:

```
curl -XGET 'localhost:9200/index1,index2/my-type/_percolate' -d '{
  "doc" : {
    "title" : "Coffee percolator",
    "body" : "A coffee percolator is a type of ..."
  }
}'
```

Aliases:

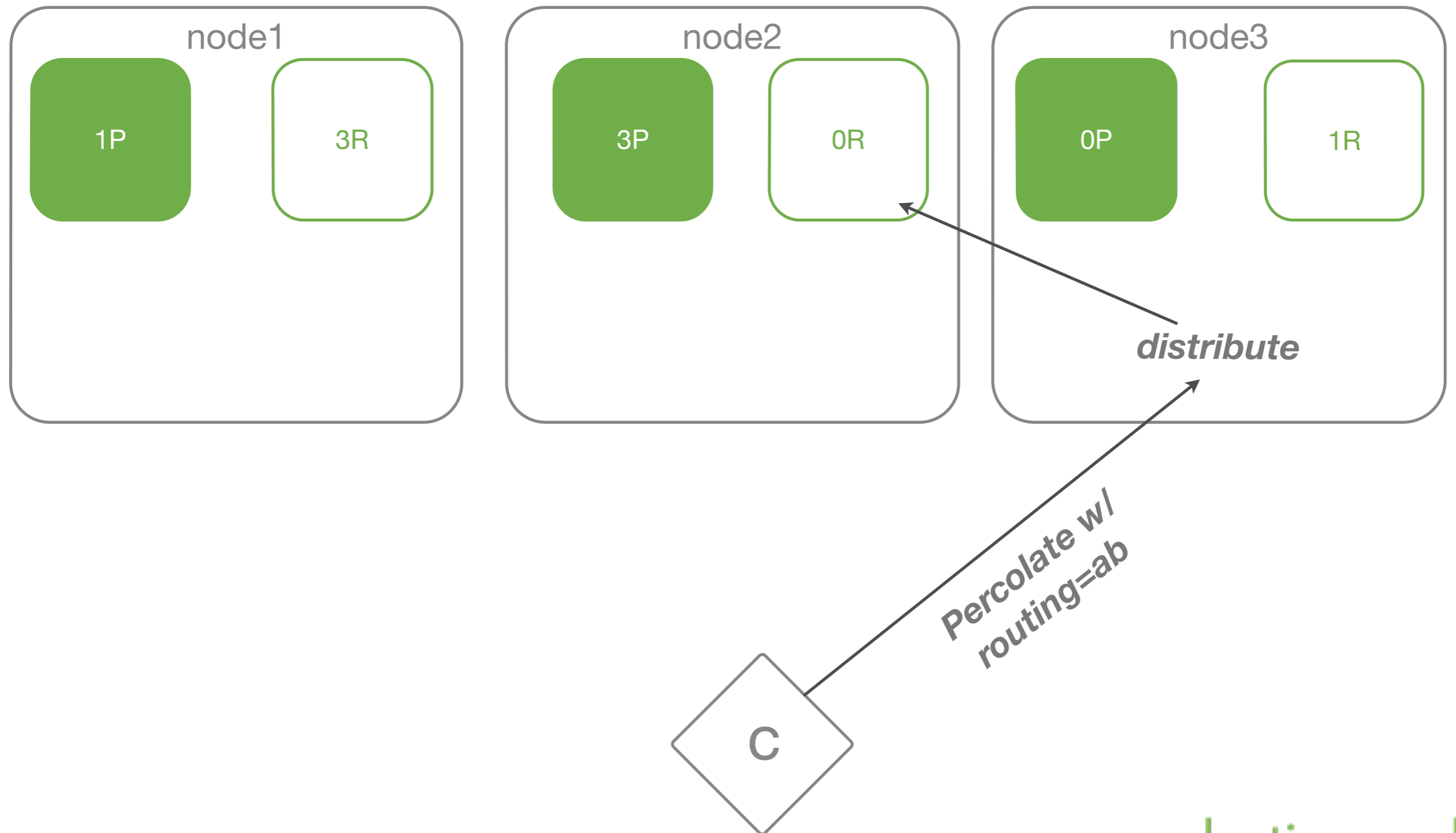
```
curl -XGET 'localhost:9200/my-alias/my-type/_percolate' -d '{
  "doc" : {
    "title" : "Coffee percolator",
    "body" : "A coffee percolator is a type of ..."
  }
}'
```

Percolator

Routing:

```
curl -XPUT 'localhost:9200/index/type/_percolate?routing=ab' -d '{
  "doc" : {
    "title" : "Coffee percolator",
    "body" : "A coffee percolator is a type of ..."
  }
}'
```

Distributed execution with routing



Percolator type

- The `.percolator` type is a hidden type.
- The search api omits **.percolator** typed documents from the response.

By default search api ignores percolator queries:

```
curl -XGET 'localhost:9200/index/_search' -d '{  
  ...  
'
```

Unless specifically specified:

```
curl -XGET 'localhost:9200/index/.percolator/_search' -d '{  
  ...  
'
```

Percolator type

- The percolator ignores the top **query** field.

Default .percolator mapping:

```
{
  "properties" : {
    "query" : {
      "type" : "object",
      "enabled" : false
    }
  }
}
```

Percolator metadata

- A percolator query is just a document!
That gets treated differently than a regular document.
- No restriction on other fields:


```
curl -XPUT 'localhost:9200/my-index/.percolator/my-id' -d '{
  "query" : {
    "match" : {
      "body" : "coffee"
    }
  },
  "organization_id" : "xyz"
}'
```


Percolator metadata

- Metadata fields do get indexed.
and can be used in percolator features.

Percolator feature filtering:

```
curl -XGET 'localhost:9200/my-index/my-type/_percolate' -d '{
  "doc" : {
    "title" : "Coffee percolator",
    "body" : "A coffee percolator is a type of ..."
  },
  "filter" : {
    "term" : {"organization_id" : "xyz"}
  }
}'
```



Only percolator queries with the specified metadata will evaluate the document being percolated.

Percolator - tips

- Percolating is CPU intensive.

Try to reduce the number of queries to be evaluated

- Let percolator queries co-exist in the same index.

If the number of queries are small

- Dedicated percolator index.

Dedicated sharding configurations

Dedicated percolator nodes

Percolator features

Feature - count api

Percolate count api:

```
curl -XPUT 'localhost:9200/my-index1/my-type/_percolate/count' -d '{
  "doc" : {
    "title" : "Coffee percolator",
    "body" : "A coffee percolator is a type of ..."
  }
}'
```

Response:

```
{
  "took" : 8,
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "count" : 5
}
```

Feature - filtering

Filter by query:

```
curl -XGET 'localhost:9200/index/my-type/_percolate/count' -d '{
  "doc" : {
    "title" : "Coffee percolator",
    "body" : "A coffee percolator is a type of ..."
  },
  "query" : {
    "term" : {"click_id" : "43"}
  }
}'
```

Feature - sorting / scoring

- Build on top on the query support.
- Sorting based on percolator query fields.
Document being percolated isn't scored!
- Three new options:
 - size The amount of matches to return (required with sort)
 - sort Whether to sort based on query.
 - score Just include score, but don't sort
- Like the query / filter support not realtime.

Feature - sorting / scoring

```
curl -XGET 'localhost:9200/my-index/my-type/_percolate' -d '{
  "doc" : {
    ...
  },
  "query" : {
    "function_score" : {
      "query" : { "match_all": {}},
      "functions" : [
        {
          "exp" : {
            "create_date" : {
              "reference" : "2014/05/26",
              "scale" : "1000d"
            }
          }
        }
      ]
    }
  }
  "sort" : true, "size" : 10
}'
```

Feature - sorting / scoring

```
{
  "took": 2,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "total": 2,
  "matches": [
    {
      "_index": "my-index",
      "_id": "2",
      "_score": 0.85559505
    },
    {
      "_index": "my-index",
      "_id": "1",
      "_score": 0.4002574
    }
  ]
}
```


Feature - aggregations

- Aggregation support on query metadata:

```
curl -XGET 'localhost:9200/my-index/my-type/_percolate' -d '{
  "doc" : {
    ""
  },
  aggs : {
    "click_ids" : {
      "terms" : {
        "field" : "click_id"
      }
    }
  }
}'
```

Feature - highlighting

Lets add two queries:

```
curl -XPUT 'localhost:9200/my-index/_percolator/1' -d '{
  "query": {
    "match" : {
      "body" : "brown fox"
    }
  }
}'
```

```
curl -XPUT 'localhost:9200/my-index/_percolator/2' -d '{
  "query": {
    "match" : {
      "body" : "lazy dog"
    }
  }
}'
```

Feature - highlighting

```
curl -XGET 'localhost:9200/my-index/my-type/percolate' -d '{
  "doc" : {
    "body" : "The quick brown fox jumps over the lazy dog"
  },
  "highlight" : {
    "fields" : {
      "body" : {}
    }
  },
  "size" : 5
}'
```

- The size option is required.
- All highlight options are supported.

Feature - highlighting

```
{
  ...
  "total": 2,
  "matches": [
    {
      "_index": "my-index",
      "_id": "1",
      "highlight": {
        "body": [
          "The quick <em>brown</em> <em>fox</em> jumps over the lazy dog"
        ]
      }
    },
    {
      "_index": "my-index",
      "_id": "2",
      "highlight": {
        "body": [
          "The quick brown fox jumps over the <em>lazy</em> <em>dog</em>"
        ]
      }
    }
  ]
}
```

x§

- Combine multiple percolate requests into a single request

Request:

```
curl -XGET 'localhost:9200/_mpercolate' --data-binary @requests.txt; echo
```

requests.txt:

```
{"percolate" : {"index" : "my-index", "type" : "my-tweet"}}  
{"doc" : {"title" : "coffee percolator"}}  
{"percolate" : {"index" : "my-index", "type" : "my-type", "id" : "1"}}  
{  
{"count" : {"index" : "my-index", "type" : "my-type"}}  
{"doc" : {"title" : "coffee percolator"}}  
{"count" : {"index" : "my-index", "type" : "my-type", "id" : "1"}}  
}
```