

Deploying Apache Gora as a Query Broker

Lewis John McGibbney
ApacheCon-NA 2014



Me, Myself and I



Con's

MemStore cannot be shared across multiple JVM's. We cannot share memory.
- Given that "map" is static then it will be shared by many MemStore within a JVM. The key may or may not be same type, which in this case the code: startKey = (K) map.getKey(); could throw exception for illegal casting.
<https://issues.apache.org/jira/browse/GORA-228>

Resource Selection Algorithm

```
int TopK = 10;
Map mObj = new HashMap();
for (int i = 0; i < TopK; i++) {
    for (Map map : maps) {
        if (map.containsKey(key)) {
            int val = map.get(key);
            if (val > mObj.get(key)) {
                mObj.put(key, val);
            }
        }
    }
}
return mObj;
```

Fundamentals

In the beginning...



Secret Store: MemStore

```
ConcurrentMap<String, Map>
exp. av log() put(k, v, T obj)
exp. av log() put(k, v, String[] fields)
exp. av log() delete(k, key)
deleteByQuery(Query q, T obj)
execute(Query q, T obj)
```

MemStore API

Architecture/Deployment



Gora as a Query Broker Approach

Focus on Resource Selection and Result Merging
Simulate geographically distributed data in heterogeneous storage mediums
Utilize a depth of Gora: Access data regardless of its location/persistent location as well as geographically

Discussion



The End

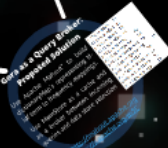
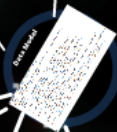
A huge thank you for the last 40 or so minutes.
Enjoy the rest of ApacheCon and your time in Denver.
lewis@apache.org
<https://twitter.com/lmcgibbney>

Apache Gora



Future Aim

Improve upon this implementation and integrate it as an example in Gora trunk.
Gora REST API so that applications can call the Query Broker.





Me, Myself and I



THE HISTORY OF THE
SCOTTISH NATION
BY JAMES H. BURNETT
1890

A bit about myself

- PostDoc Stanford University, CA '13 - '14
Engineering Informatics
- PhD Candidate Glasgow Caledonian University,
UK '09 - '12 Legislative Informatics
- <'09 ...Quantity Surveyor!!!
- Interests:
 - Search Engines, Web Search, NoSQL, Open
Data (probably in that order)
- Analyzing and addressing REAL problems.
- Apache Member, Nutch PMC, Gora PMC, Any23
PMC, OODT PMC, Apache TAC, Usergrid
(incubating) PPMC



Agenda

- The Story Begins... Invisible Engines
- Domain of Application: Federated Web Search
- The TREC FWS Track
- FWS Theory
- Apache Gora
- Gora as a Query Broker
- Approach/Solution/Architecture
- Discussion/Questions

Feel free to ask questions as we go through

<https://github.com/lewismc/queryBroker>

In the beginning...





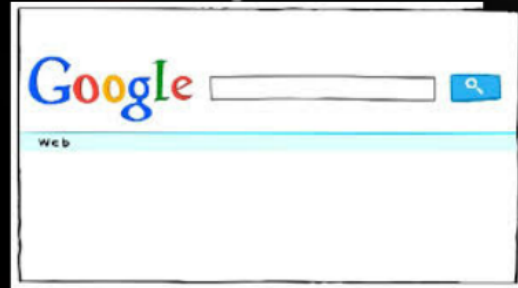
local/national trade

Multi/International





Global: eCommerce



What do they all have in common?

Invisibility can be a strength...





In this case totally invisible!!!

Web Images Au

Search



INVISIBLE ENGINES

How Software Platforms
Drive Innovation
and Transform Industries

David S. Evans
Andrei Hagiu
Richard Schmalensee

Some points to ponder over as we move on

- 1 - In 2008 it was easy to equate Windows OS and the PC industry (70% 2008 to about 30% in 2012*).
- 2 - It is however less trivial to equate Android OS and the mobile market** as Android is only one element in a complex structure that links mobile phone operators, handset makers, application providers, and software platform makers.

Quote of the day

"underlying software platform technology shape these industries, and the business strategies employed by firms in those industries, in fundamental and important ways"

Apache Web Server.. Apache Hadoop... _insert_next_game_changer

*According to Forrester Research

 **Android >80% market share: 211.6m units in 3Q13

Federated Web Search

Federated search is an information retrieval methodology that allows the simultaneous search of multiple searchable resources. A user makes a single query request which is distributed to the search engines participating in the federation. The federated search then aggregates the results that are received from the search engines for presentation to the user.



Federated search has the potential of improving web search: the user becomes less dependent on a single search provider and parts of the deep web become available through a unified interface, leading to a wider variety in the retrieved search results.

Typical comparison sites are a good example of how FWS can well serve a specific purpose(s) however consider the following:

Observation:

From `_most_` of the sites currently available, it would seem that for each query, ALL data sources are involved within the federated query process. Assume that query results are returned as an unorganized list of data (based on faster responses entering the list earlier), linear time ($\Omega(n)$) (lower bound) is required to find the minimum element. Add on variables relating 3rd party response times, etc. this explains why we have to wait for query execution and subsequent presentation of results. So in fact running time is much less efficient than ($\Omega(n)$).

Although the user cannot then query the results, if we consider sorting the list/array, in which case only one initial, expensive sort is needed, followed by many cheap selection operations we would obtain $O(1)$ (upper bound) for an array, though selection is $O(n)$ in a list, even if sorted, due to lack of random access. In general, sorting requires $O(n \log n)$ time, where n is the length of the list.

Question(s):

Is there a better way of doing this?

Do we need to query **EVERY** underlying data source **EVERY** time?

What happens when we are dealing with domains other than price comparisons that work with Integer's?

TREC FWS Track

The track investigates techniques for the selection and combination of search results from a large number of real on-line web search services. A list of 157 search engines is made available with sampled search results from each of these engines.

Task1: Vertical Selection

In web search, a vertical is associated with content dedicated to either a host (e.g. "finance"), a media type (e.g. "images") or a genre (e.g. "news"). For example, an "image" vertical contains resources such as Flickr and Picasa.

Therefore, the system should select a subset of verticals to retrieve from.

Input: A query

Output: A set of relevant verticals

Evaluation: Based on standard classification metrics. Participants train models, produce and recall. The set of relevant verticals will be judged on the relevance of the individual search results provided by the resources in that vertical.

Task2: Resource Selection

For practical reasons, it is not possible to query all available resources (search engines) when a query is issued to a federated search system. Therefore, the system first needs to select the appropriate search engines for the given query.

For example, suitable resources for a query such as "Pittsburgh Steelers News" might be ESPN, Fox Sports, etc. To simulate a realistic setting, the participants are not allowed to sample or retrieve results from the resources themselves.

Input: A query

Output: A ranking of resources (the most appropriate resources are ranked highest)

Evaluation: The relevance of each resource is determined by calculating the graded precision on its top 10 results.

*Using graded relevance assessments in IR evaluation, J. Kekäläinen and K. Järvelin, JASIST 53(3), 2002

Task3: Results Merging

The goal of results merging is to merge the search result snippets from previously selected resources in a single ranked list similar to that which we see in our price comparison sites, etc.

Input: A query

Output: A ranking of resources (the most appropriate resources are ranked highest)

Evaluation: Using two metrics: nDCG* to measure topical relevance, and IR-nDCG to measure diversity between verticals in addition to topical relevance.

*Christopher Burges et al., Learning to rank using gradient descent, ICML 2005.

<https://sites.google.com/site/trecfedweb/>

Task1: Vertical Selection

In web search, a vertical is associated with content dedicated to either a topic (e.g. “finance”), a media type (e.g. “images”) or a genre (e.g. “news”). For example, an “image” vertical contains resources such as Flickr and Picasa.

Therefore, the system should select a subset of verticals to retrieve from.

Input: A query

Output: A set of relevant verticals

Evaluation: Based on standard classification metrics: F-measure (main metric), precision and recall. The set of relevant verticals will be based on the relevance of the individual search results provided by the resources in that vertical.

Task2: Resource Selection

For practical reasons, it is not possible to query all available resources (search engines) when a query is issued to a federated search system. Therefore, the system first needs to select the appropriate search engines for the given query.

For example, suitable resources for a query such as 'Pittsburgh Steelers News' might be ESPN, Fox Sports, etc. To simulate a realistic setting, the participants are not allowed to sample or retrieve results from the resources themselves.

Input: A query

Output: A ranking of resources (the most appropriate resources are ranked highest)

Evaluation: The relevance of each resource is determined by calculating the graded precision* on its top 10 results.

*Using graded relevance assessments in IR evaluation, J. Kekäläinen and K. Järvelin, JASIST 53(13), 2002

Task3: Results Merging

The goal of results merging is to merge the search result snippets from previously selected resources in a single ranked list similar to that which we see in our price comparison sites, etc.

Input: A query

Output: A ranking of resources (the most appropriate resources are ranked highest)

Evaluation: Using two metrics: nDCG* to measure topical relevance, and IA-nDCG to measure diversity between verticals in addition to topical relevance.

*Christopher Burges et al. (Learning to rank using gradient descent. ICML 2005).

Apache Gora

Generic Object Representation using Avro

The Apache Gora open source framework provides an in-memory data model and persistence for big data. Gora supports persisting to column stores, key value stores, document stores and RDBMSs, and analyzing the data with extensive Apache Hadoop™ MapReduce support.

Several data stores supported



DataStore Support



2.0.X



1.3.12



4.3.0



1.5.X



4.3.0



0.94.14



1.0.1



1.7.X



2.0.39 (client driver)

Secret Store: MemStore

ConcurrentSkipListMap:

exp. $av \log(n)$

put(K key, T obj)

exp. $av \log(n)$

get(K key, String[] fields)

exp. $av \log(n)$

delete(K key)

deleteByQuery(Query<K, T> query)

```
public Result<K, T> execute(Query<K, T> query) {  
    K startKey = query.getStartKey();  
    K endKey = query.getEndKey();  
    if (startKey == null) {  
        startKey = (K) map.firstKey();  
    }  
    if (endKey == null) {  
        endKey = (K) map.lastKey();  
    }  
    //check if query.fields is null  
    query.setFields(getFieldsToQuery(query.getFields()));  
    ConcurrentNavigableMap<K, T> submap = map.subMap(startKey, true, endKey, true);  
    return new Result<K, T>(this, query, submap);  
}
```

- map.firstKey() = $O(1)$ constant
- map.lastKey() = $O(n)$ linear unless map is modified
- If no fields are requested, we get ALL fields
- Create a map.subMap first and last keys inclusive
- return Result(K, T)

execute(Query<K, T> query)



```

public Result<K, T> execute(Query<K, T> query) {
    K startKey = query.getStartKey();
    K endKey = query.getEndKey();
    if (startKey == null) {
        startKey = (K) map.firstKey();
    }
    if (endKey == null) {
        endKey = (K) map.lastKey();
    }

    //check if query.fields is null
    query.setFields(getFieldsToQuery(query.getFields()));

    ConcurrentNavigableMap<K, T> submap = map.subMap(startKey, true, endKey, true);

    return new MemResult<K, T>(this, query, submap);
}

```

- `map.firstKey()` = $O(1)$ constant
- `map.lastKey()` = $O(n)$ linear unless map is modified
- If no fields are requested, we get ALL fields
- Create a `map.subMap` first and last keys inclusive
- return `Result(K, T)`

Gora as a Query Broker: Approach

Focus on **Resource Selection and Results Merging**

Simulate geographically distributed data in heterogeneous storage mediums

Utilize a strength of Gora: Access data regardless of its location (persistent location as well as geographical)

Gora as a Query Broker: Proposed Solution

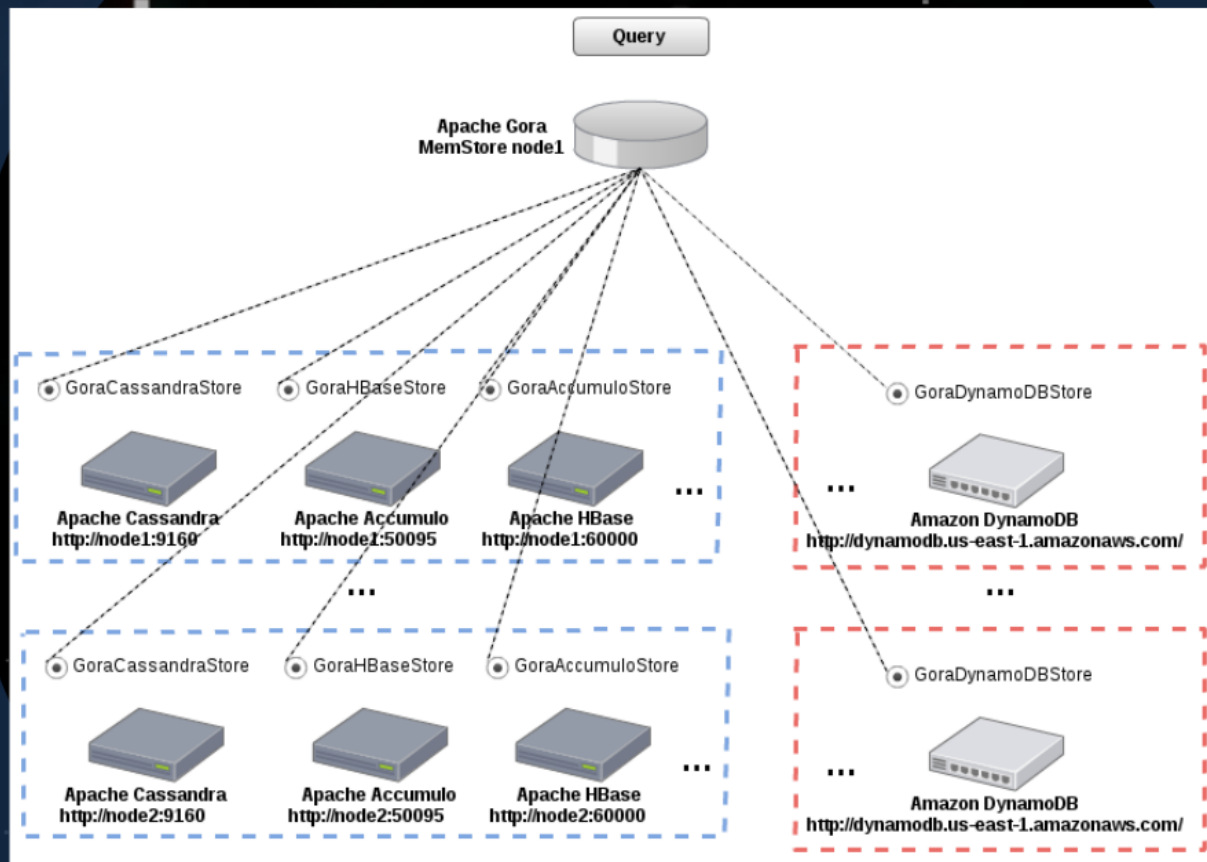
Use Apache Mahout* to build dictionaryMap's representing tf-idf term to frequency mappings.

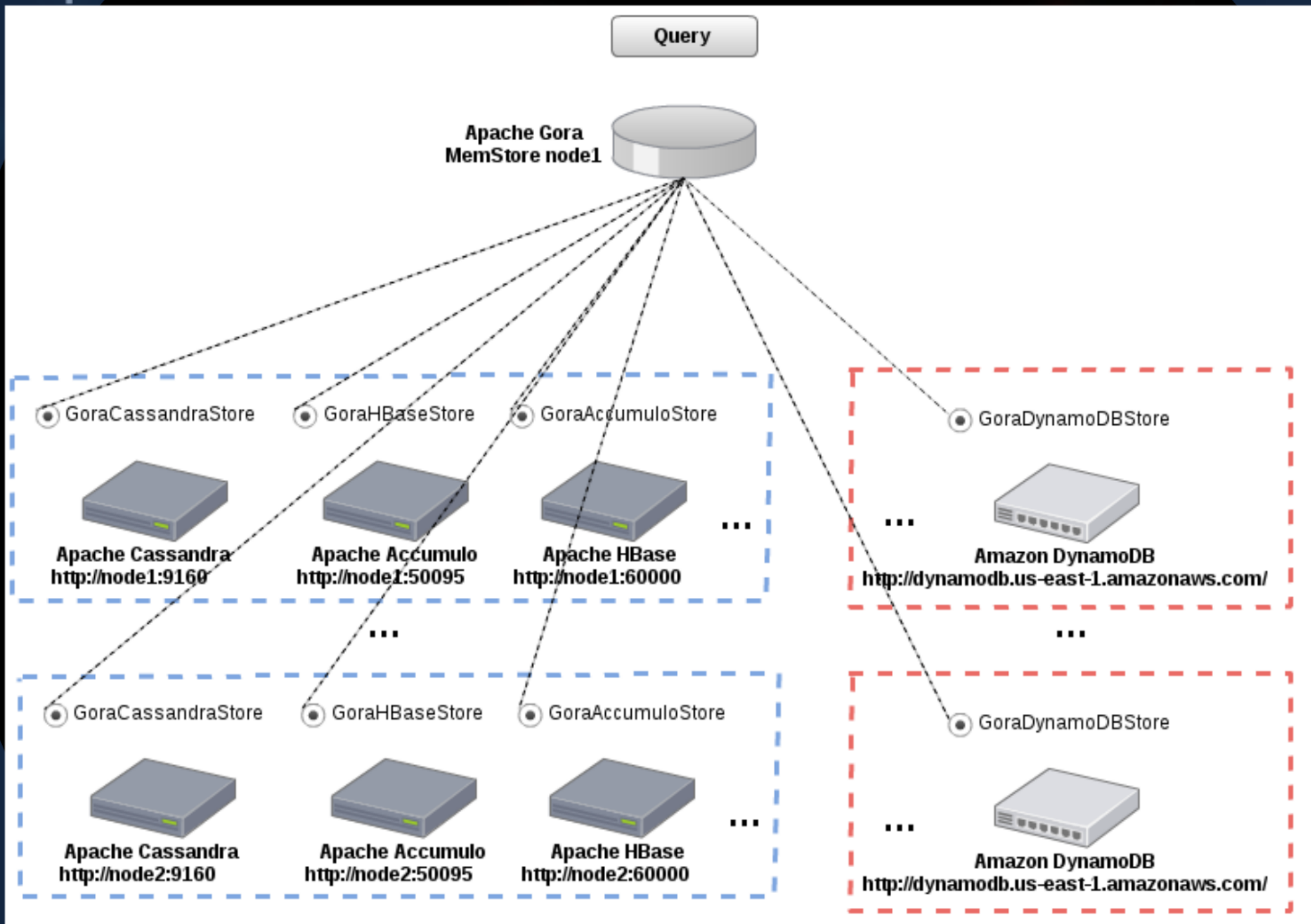
Use MemStore as a cache and as a broker between incoming queries and data store selection

6738	encode	21
6739	encoded	27
6740	encoder	10
6741	encodes	18
6742	encoding	42
6743	encompass	16
6744	encompasses	18
6745	encompassing	
6746	encounter	34
6747	encountered	34
6748	encounters	37
6749	encourage	16

*<http://mahout.apache.org>
<http://s.apache.org/m5C>

Architecture/Deployment





Data Model

```
{
  "name": "DataStore", "default": null,
  "type": "record",
  "doc": "Record containing Fields required to build a Query Broker",
  "namespace": "org.lewismc.store",
  "fields": [
    {
      "name": "goraDataStoreName", "type": ["null", "string"],
      "default": null, "doc": "The Gora DataStore reference"},
    {
      "name": "nativeDataStoreName", "type": ["null", "string"],
      "default": null, "doc": "The native DataStore identifier"},
    {
      "name": "_ipAddress", "type": ["null", "string"],
      "default": null, "doc": "The native DataStore IP Address"},
    {
      "name": "nativeDataStoreVersion", "type": ["null", "string"],
      "default": null, "doc": "The native DataStore version"},
    {
      "name": "dataSize", "type": "long",
      "default": 0, "doc": "The data volume in this particular DataStore"},
    {
      "name": "numberOfRecords", "type": "int",
      "default": 0, "doc": "The number of individual objects in this DataStore"},
    {
      "name": "dictionaryMap", "type": {"type": "map", "values": "string"},
      "default": {}, "doc": "A Map containing token --> tf-idf mappings"}
  ]
}
```

Resource Selection Algorithm

```
int TOP_K = 10;
Map simObjs = new HashMap();
int count = 0;
for (Map map : maps) {
    for (String term: terms){
        if (map.containsKey(term)) {
            int freq = map.get(term)
            if (freq > 0) {
                higher = freq;
                simObjs.put(resourceName , map);
                count++;
            }
        }
    } if (count < TOP_K)
        break;
}
```

Con's

- MemStore cannot be shared across multiple JVM's. We cannot share memory.
- Given that "map" is static then it will be shared by many MemStore within a JVM. The Key may or may not be same types, which in this case, the code: `startKey = (K) map.firstKey();` could throw exception for illegal casting.
- <https://issues.apache.org/jira/browse/GORA-228>

Future Aim

- Improve upon this implementation and integrate it as an example in Gora trunk.
- Gora REST API so that applications can call the Query Broker.

Discussion



The End



**A huge thank you for the last 40 or so
minutes.
Enjoy the rest of ApacheCon and your time
in Denver**

**dev/user-subscribe@gora.apache.org
lewismc@apache.org
@hectorMcSpector**