

# NoSQL Cocktail Hour\*

Having Awkward Conversations with Some Popular Modern Non-Relational  
Data Processing Systems

John Haldeman @ Hackforge – August 2016

\* Apologies If I Run Over the Hour

# NoSQL Cocktail Hour

**For These Systems:**



*cassandra*



**Talk About the following:**

1. General Architecture
2. Origins
3. Do “Something” (Use each to answer a question)
4. Security
5. Where it’s used and potential drawbacks
6. Similar Systems

# But First – The CAP Theorem

- You can only guarantee two of the following:
  - Consistency (AKA Atomic Consistency): The client can treat the data as a *single* up-to-date *copy*
  - Availability: All clients can access (update) the data at any time
  - Partition Tolerance: The system runs well when it's distributed across network partitions
- Modern DBMSs are partitioned systems, so the big tradeoffs are in C and A (note in a non-network partitioned system, you can have both C and A)
- [https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed:](https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed)
  - The easiest way to understand CAP is to think of two nodes on opposite sides of a partition:
  - Allowing at least one node to update state will cause the nodes to become inconsistent, thus forfeiting C
  - Likewise, if the choice is to preserve consistency, one side of the partition must act as if it is unavailable, thus forfeiting A
  - Only when nodes communicate is it possible to preserve both consistency and availability, thereby forfeiting P
- It's not black and white and systems can behave differently depending on circumstance (see "Cap-latency connection" side bar of article above)
- This looks like a good resource: <http://blog.nahurst.com/visual-guide-to-nosql-systems>



# General Architecture

Store JSON Documents

Protocol based on a Binary Version of JSON called BSON

No Schemas

Scales Via Sharding (Horizontal Partitioning)

Replica Sets for High Availability

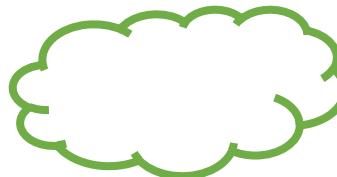
CAP: Values Consistency and Partition Tolerance over Availability

```
{  
    "address": {  
        "building": "1007",  
        "coord": [ -73.856077, 40.848447 ],  
        "street": "Morris Park Ave",  
        "zipcode": "10462"  
    },  
    "borough": "Bronx",  
    "cuisine": "Bakery",  
    "grades": [  
        { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },  
        { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },  
        { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },  
        { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },  
        { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }  
    ],  
    "name": "Morris Park Bake Shop",  
    "restaurant_id": "30075445"  
}
```

More info at: <https://www.mongodb.com/mongodb-architecture>

# Origins

- 10gen
  - Former DoubleClick employees get together to build a Platform as a Service (PaaS) provider
  - They didn't like the databases available, started building mongoDB
  - 10gen's PaaS business never took off, but mongoDB did well. 10gen eventually became MongoDB, Inc.



# Do “Something” With MongoDB



- First we need data. How about this:

<http://open.canada.ca/data/en/dataset/00a331db-121b-445d-b119-35dbe3eudd9>

- National broadband Data (2012 – a little out of date, but good for an example)
- Let’s see if we can find any places in Windsor/Essex that are unserved/underserved by broadband (< 1.5 Mbps available)

The screenshot shows the Government of Canada's Open Data portal. The top navigation bar includes links for Jobs, Immigration, Travel, Business, Benefits, Health, and Tax. The main content area is titled "NATIONAL BROADBAND DATA". It describes the dataset as representing broadband coverage information by technology for existing broadband service providers. Coverage information for the Broadband Canada Program is included for completed projects. The data is aggregated over a grid of hexagons, each 6 km across, with estimated ranges for unserved and underserved populations. A "Licence" section points to the "Open Government Licence - Canada". Below that, a "Dataset Resources" table lists the "Dataset" with "Format" as "CSV" and "Language" as "English". A "Download" button is located at the bottom right of the table.

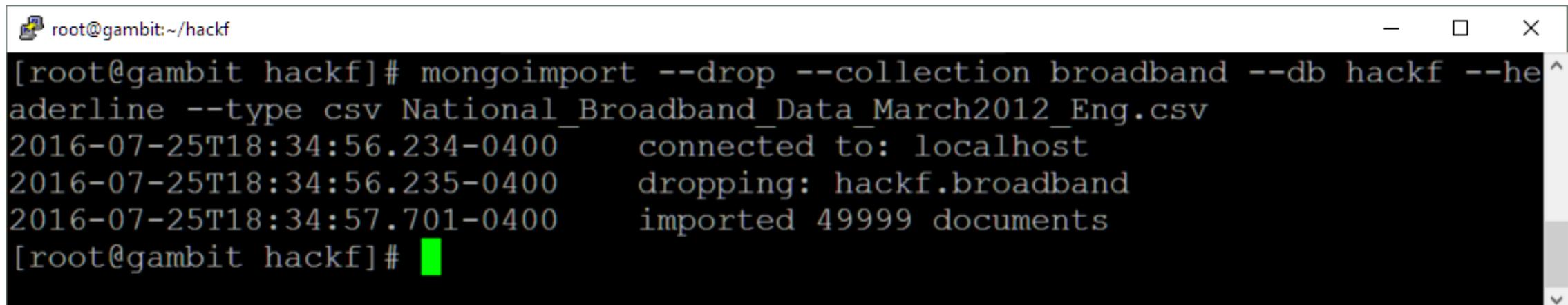
Resource Name	Format	Language	Link
Dataset	CSV	English	<button>Download</button>

# Do “Something” With MongoDB



## Import CSV file:

```
mongoimport --drop --collection broadband --db hackf --headerline --type csv National_Broadband_Data_March2012_Eng.csv
```

A screenshot of a terminal window titled "root@gambit:~/hackf". The window contains the command "mongoimport --drop --collection broadband --db hackf --headerline --type csv National\_Broadband\_Data\_March2012\_Eng.csv" and its execution output. The output shows the command being run, followed by connection details, the collection being dropped, and the number of documents imported (49999).

```
[root@gambit hackf]# mongoimport --drop --collection broadband --db hackf --headerline --type csv National_Broadband_Data_March2012_Eng.csv  
2016-07-25T18:34:56.234-0400    connected to: localhost  
2016-07-25T18:34:56.235-0400    dropping: hackf.broadband  
2016-07-25T18:34:57.701-0400    imported 49999 documents  
[root@gambit hackf]#
```

# Do “Something” With MongoDB



What does the data look like. Get first three rows of collection using the MongoDB command line client (mongo):

```
mongo
```

```
use hackf
```

```
db.broadband.find().limit(3);
```

# Do “Something” With MongoDB



- What does the data look like:

```
[root@gambit hackf]# mongo
MongoDB shell version: 3.2.9-rc0
connecting to: test
Server has startup warnings:
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten]
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten] ** WARNING: /sys/kerne
l/mm/transparent_hugepage/enabled is 'always'.
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten] ** We suggest s
etting it to 'never'
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten]
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten] ** WARNING: /sys/kerne
l/mm/transparent_hugepage/defrag is 'always'.
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten] ** We suggest s
etting it to 'never'
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten]
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten] ** WARNING: soft rlimi
ts too low. rlimits set to 4096 processes, 64000 files. Number of processes sh
ould be at least 32000 : 0.5 times number of files.
2016-07-25T18:31:29.677-0400 I CONTROL [initandlisten]
> use hackf
switched to db hackf
> db.broadband.find().limit(3);
{
  "_id" : ObjectId("57969410fc6774f87a34aa5e"),
  "Hexagon Number" : 40930,
  "GSA
Number" : "",
  "First Nation" : "",
  "Location Name" : "Aalders Landing, NS @ 4
4.82□N x 64.94□W",
  "Municipality" : "Annapolis, Subd. D :SC",
  "Latitude" : 44.
82,
  "Longitude" : -64.94,
  "Total Population 2006 Census" : 154,
  "Unserved / Un
derserved Population in Hexagon" : 0,
  "Deferral Account" : "F",
  "DSL Available
" : "F",
  "Cable Available" : "F",
  "Wireless Available" : "T"
}
{
  "_id" : ObjectId("57969410fc6774f87a34aa5f"),
  "Hexagon Number" : 41100,
  "GSA
Number" : "",
  "First Nation" : "",
  "Location Name" : "Abercrombie, NS @ 45.6□
N x 62.71□W",
  "Municipality" : "Pictou, Subd. B :SC",
  "Latitude" : 45.6,
  "Long
itude" : -62.71,
  "Total Population 2006 Census" : 447,
  "Unserved / Underserved
Population in Hexagon" : 0,
  "Deferral Account" : "F",
  "DSL Available" : "T",
  "Cable Available" : "T",
  "Wireless Available" : "T"
}
{
  "_id" : ObjectId("57969410fc6774f87a34aa60"),
  "Hexagon Number" : 40049,
  "GSA
Number" : "",
  "First Nation" : "",
  "Location Name" : "Abercrombie, NS @ 45.65
□N x 62.71□W",
  "Municipality" : "Pictou, Subd. B :SC",
  "Latitude" : 45.65,
  "Lo
ngitude" : -62.71,
  "Total Population 2006 Census" : 582,
  "Unserved / Underserv
ed Population in Hexagon" : 0,
  "Deferral Account" : "F",
  "DSL Available" : "T"
,
  "Cable Available" : "T",
  "Wireless Available" : "T"
}
> █
```

# Do “Something” With MongoDB



- What does the data look like (pretty)
- Completely flat JSON objects. Fine for our example, but if you are using MongoDB to develop an app, it's likely to have more structure

```
root@gambit:~/hackf
> db.broadband.find().limit(3).pretty();
{
    "_id" : ObjectId("57969410fc6774f87a34aa5e"),
    "Hexagon Number" : 40930,
    "GSA Number" : "",
    "First Nation" : "",
    "Location Name" : "Aalders Landing, NS @ 44.82°N x 64.94°W",
    "Municipality" : "Annapolis, Subd. D :SC",
    "Latitude" : 44.82,
    "Longitude" : -64.94,
    "Total Population 2006 Census" : 154,
    "Unserved / Underserved Population in Hexagon" : 0,
    "Deferral Account" : "F",
    "DSL Available" : "F",
    "Cable Available" : "F",
    "Wireless Available" : "T"
}
{
    "_id" : ObjectId("57969410fc6774f87a34aa5f"),
    "Hexagon Number" : 41100,
```

# Do “Something” With MongoDB



## Get Windsor-Essex Municipalities with a regex:

```
db.broadband.find({  
    "Municipality":/^Windsor:CY|^Essex.*|^Amherstburg.*|^LaSalle.*|^Tecumseh.*|^Lakeshore.*/  
}  
, {  
    _id : 0,  "Municipality" : 1,  "Location Name" : 1  
});
```

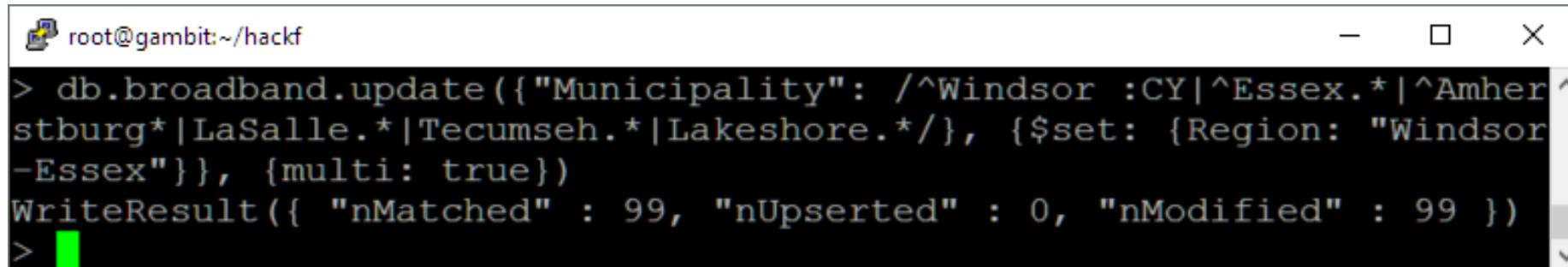
```
root@gambit:~/hackf  
> db.broadband.find({"Municipality": /^Windsor :CY|^Essex.*|^Amherstburg*|LaSalle.*|Tecumseh.*|Lakeshore.*/}, { _  
id: 0, "Municipality": 1, "Location Name": 1});  
{ "Location Name" : "Amherstburg, ON @ 42.09N x 83.12W", "Municipality" : "Amherstburg :T" }  
{ "Location Name" : "Amherstburg, ON @ 42.11N x 83.08W", "Municipality" : "Amherstburg :T" }  
{ "Location Name" : "Amherstburg, ON @ 42.13N x 83.12W", "Municipality" : "Amherstburg :T" }  
{ "Location Name" : "Arner, ON @ 42.02N x 82.84W", "Municipality" : "Essex :T" }  
{ "Location Name" : "Barretville, ON @ 42.11N x 82.84W", "Municipality" : "Essex :T" }  
{ "Location Name" : "Auld, ON @ 42.13N x 83.02W", "Municipality" : "Amherstburg :T" }  
{ "Location Name" : "Belle River, ON @ 42.29N x 82.72W", "Municipality" : "Lakeshore :T" }  
{ "Location Name" : "Busy Bee Corners, ON @ 42.09N x 83.02W", "Municipality" : "Amherstburg :T" }  
{ "Location Name" : "Colchester, ON @ 41.97N x 82.96W", "Municipality" : "Essex :T" }  
{ "Location Name" : "Colchester, ON @ 42N x 82.9W", "Municipality" : "Essex :T" }  
{ "Location Name" : "Comber, ON @ 42.22N x 82.54W", "Municipality" : "Lakeshore :T" }  
{ "Location Name" : "Deerbrook, ON @ 42.29N x 82.6W", "Municipality" : "Lakeshore :T" }  
{ "Location Name" : "Delisle's Corners, ON @ 42.18N x 83.02W", "Municipality" : "Amherstburg :T" }  
{ "Location Name" : "Edgewater Beach, ON @ 42.18N x 83.12W", "Municipality" : "Amherstburg :T" }  
{ "Location Name" : "Elmstead, ON @ 42.29N x 82.84W", "Municipality" : "Lakeshore :T" }  
{ "Location Name" : "Essex, ON @ 41.97N x 82.84W", "Municipality" : "Essex :T" }  
{ "Location Name" : "Essex, ON @ 42.15N x 82.84W", "Municipality" : "Essex :T" }  
{ "Location Name" : "Essex, ON @ 42.18N x 82.78W", "Municipality" : "Essex :T" }  
{ "Location Name" : "Essex, ON @ 42.2N x 82.84W", "Municipality" : "Lakeshore :T" }  
{ "Location Name" : "Fairplay, ON @ 42.27N x 82.9W", "Municipality" : "Tecumseh :T" }  
Type "it" for more  
>
```

# Do “Something” With MongoDB



**Update Those Documents with a New Field “Region”:**

```
db.broadband.update ({
  "Municipality": /^Windsor:CY|^Essex.*|^Amherstburg.*|^LaSalle.*|^Tecumseh.*|^Lakeshore.*/ ,
  {$set: {"Region" : "Windsor-Essex"}},
  {multi : true});
```

A screenshot of a terminal window titled "root@gambit:~/hackf". The window contains a command-line session where a MongoDB update operation is performed on the "broadband" collection. The command uses a regular expression to match documents from five municipalities and adds a new "Region" field with the value "Windsor-Essex". The output shows a "WriteResult" object with "nMatched" set to 99, "nUpserted" to 0, and "nModified" to 99. A green vertical bar is present on the left side of the terminal window.

```
root@gambit:~/hackf
> db.broadband.update({ "Municipality": /^Windsor:CY|^Essex.*|^Amherstburg.*|^LaSalle.*|^Tecumseh.*|^Lakeshore.*/ },
  {$set: {"Region" : "Windsor-Essex"}}, {multi: true})
WriteResult({ "nMatched" : 99, "nUpserted" : 0, "nModified" : 99 })
>
```

Notice the lack of schema modifications – We just write to a field that never existed before and it only exists for those records updated

# Do “Something” With MongoDB



MongoDB Aggregation Pipelines. Get the categories of under served areas and calculate some stats:

```
db.broadband.aggregate([
    {
        $match: {Region: "Windsor-Essex"} },
        {$group:
            { id: "$Unserved / Underserved Population in Hexagon",
                num: { $sum: 1},
                pop: {$sum: "$Total Population 2006 Census"} }
        }
    ]
);
```

# Do “Something” With MongoDB



MongoDB Aggregation Pipelines. Get the categories of under served areas and calculate some stats:

```
root@gambit:~> db.broadband.aggregate([{$match: {Region: "Windsor-Essex"}}, {$group: {_id: "$Unserved / Underserved Population in Hexagon", num: { $sum: 1}, pop: {$sum: "$Total Population 2006 Census"} }}]);  
{ "_id" : "400 - 799", "num" : 1, "pop" : 1575 }  
{ "_id" : 0, "num" : 98, "pop" : 342066 }  
>  
>  
> [REDACTED]
```

# Do “Something” With MongoDB



Add in a \$push to get the region names in each category:

```
db.broadband.aggregate(  
  [  
    {  
      $match: {Region: "Windsor-Essex"},  
      {$group:  
        {_id: "$Unserved / Underserved Population in Hexagon",  
         num: { $sum: 1},  
         pop: { $sum: "$Total Population 2006 Census"},  
         areas: { $push: {Municipality: "$Municipality", Location: "$Location Name" } }  
      }  
    }  
  ]  
);
```

# Do “Something” With MongoDB



Add in a \$push to get the region names in each category:

```
root@gambit:~
> db.broadband.aggregate([{$match:{Region: "Windsor-Essex"}}, {$group: {_id: "$Unserved / Underserved Population in Hexagon", num: { $sum: 1}, pop: {$sum: "$Total Population 2006 Census"}, areas: {$push: {Municipality: "$Municipality", Location: "$Location Name" }}}}).pretty();
{
    "_id" : "400 - 799",
    "num" : 1,
    "pop" : 1575,
    "areas" : [
        {
            "Municipality" : "Lakeshore :T",
            "Location" : "Stoney Point/Pointe-aux-Roches, ON @ 42.31°N x 82.54°W"
        }
    ]
}
{
    "_id" : 0,
    "num" : 98,
    "pop" : 342066,
    "areas" : [
        {
            "Municipality" : "Amherstburg :T",
            "Location" : "Amherstburg, ON @ 42.09°N x 83.12°W"
        },
        {
            "Municipality" : "Essex County :T",
            "Location" : "Essex County, ON @ 42.31°N x 82.54°W"
        }
    ]
}
```



# Security



Feature	Notes
<b>Authentication</b>	None (default), Native, x.509, LDAP("enterprise"), Kerberos ("enterprise")
<b>Authorization (access controls)</b>	Administrative, database and collection level privileges available to assign to roles. More difficult to limit access to documents and elements within a collection. User admins can be separated from DB admins, which can be separated from users.
<b>Encryption – Client/Server Communication</b>	TLS/SSL Supported
<b>Encryption – Data on Disk</b>	Wired Tiger (newer storage engine) allows for native encryption with external management of keys in a KMIP provider
<b>Auditing</b>	Built in Facility Available

# Where it's used



Good List: <https://www.mongodb.com/who-uses-mongodb>

When I first encountered it was primarily e-commerce use cases:

craigslist

ebay

Expedia®

Now becoming common in corporate/government use cases too:

MetLife



This a great “all-rounder” that’s easy for program for and query after the fact.

# Some things it might not be good at

A lot of semi-religious debate over “why my database choice is better than yours”:

- Historical issues with concurrency control (locking) but those may have been resolved with recent storage engine updates
- Some feel it may be too memory intensive compared to alternatives like Cassandra – and fails to scale when memory runs out

But the biggest things come from architecture:

- Joins across multiple collections are pretty awkward ([although significantly easier in V3.2](#)) – this is a document database after all, not an object-relational database
- [Atomicity and transaction](#) control can be problematic - especially when you compare it against many relational DBMSs
- Feature richness may come at the cost of performance (when compared to databases like redis for transactional workloads)



# Similar Systems



CouchDB: A document database as well, storing JSON documents. Big differences:

- 1) Accessed via REST+HTTP instead of with BSON clients
- 2) Complex querying limited to map-reduce (which is also available in MongoDB). There is no aggregation pipelining
- 3) Started by Damien Katz – who used to develop for Notes/Domino, which is also a document datastore (Document datastores have been around for a long while)



- 4) Katz stopped working on CouchDB to work on Couchbase. CouchDB != Couchbase, but MongoDB, CouchDB, and CouchBase are all JSON Document datastores





# General Architecture

redis is a key-value store

In memory, with varying degrees of persistence available – Can provide zero persistence and is sometimes used as a pure cache layer

CAP: Values Consistency and Partition Tolerance over availability

Protocol (interestingly human readable instead of binary) and storage format is redis specific

No Schemas

Sharding is available (partitioning the keyspace) and master/slave replication capabilities for HA

```
root@gambit:~/redis-3.2.3
127.0.0.1:6379> get hackfkey
"Hackforge!"
127.0.0.1:6379> scan 0 match hackfk* count 100000
1) "0"
2) 1) "hackfkey"
127.0.0.1:6379> append hackfkey 'Windsor!'
(integer) 18
127.0.0.1:6379> get hackfkey
"Hackforge!Windsor!"
127.0.0.1:6379> del hackfkey
(integer) 1
127.0.0.1:6379> get hackfkey
(nil)
127.0.0.1:6379>
```



# Data Structures

In addition to String Values, the following data structures are available:

- Hash Tables
- Ordered Lists (like arrays)
- Sets (unique unordered lists)
- Ordered Sets (unique lists ordered by score)
- Publisher/Subscriber channels
- Geospatial Data



# Redis...

The only DBMS I know of that reference algorithmic time complexity in the documentation for each command...

So, yes, they are very concerned about performance over pretty much everything else

redis Commands Clients Documentation Community Download Support License

## ZRANGE key start stop [WITHSCORES]

Available since 1.2.0.

**Time complexity:**  $O(\log(N)+M)$  with  $N$  being the number of elements in the sorted set and  $M$  the number of elements returned.

Returns the specified range of elements in the sorted set stored at key. The elements are considered to be ordered from the lowest to the highest score. Lexicographical order is used for elements with equal score. See [ZREVRANGE](#) when you need the elements ordered from highest to lowest score (and descending lexicographical order for elements with equal score).

Both start and stop are zero-based indexes, where 0 is the first element, 1 is the next element and so on. They can also be negative numbers indicating offsets from the end of the sorted set, with -1 being the last element of the sorted set, -2 the penultimate element and so on.

start and stop are **inclusive ranges**, so for example ZRANGE myzset 0 1

Related commands

- [ZADD](#)
- [ZCARD](#)
- [ZCOUNT](#)
- [ZINCRBY](#)
- [ZINTERSTORE](#)
- [ZLEXCOUNT](#)
- [ZRANGE](#)
- [ZRANGEBYLEX](#)
- [ZRANGEBYSCORE](#)
- [ZRANK](#)
- [ZREM](#)
- [ZREMRANGEBYLEX](#)
- [ZREMRANGEBYRANK](#)
- [ZREMRANGEBYSCORE](#)
- [ZREVRANGE](#)

# Origins



- Salvatore Sanfilippo started the database after he had trouble storing and displaying real time web traffic data in MySQL (high write loads – read last N records)
- It was his hobby. He went to VMWare and worked on it awhile at VMWare
- That's pretty much it. The guy still heads up development for it and works at Redis Labs



# Do “Something” With Redis

- First we need data. How about this:

<http://open.canada.ca/data/en/dataset/53753f06-8b28-42d7-89f7-04cd014323b0>

- Contracts awarded by Public Works and Government Services Canada since January 2009

- Let's see what companies were awarded the most in government contracts

Government of Canada Gouvernement du Canada

Jobs ▾ Immigration ▾ Travel ▾ Business ▾ Benefits ▾ Health ▾ Taxes ▾ More services ▾

Home → Open Government → Open Data → Search Open Data

## Contract History

Contract History is information about contracts awarded by Public Works and Government Services Canada (PWGSC) since January 2009, on behalf of federal departments and agencies. The Contract History CSV files are updated monthly, and are organized by fiscal year. A complete Contract History file, containing contracts from 2009 to today is also available for download on this page. Visit the "About Contract History" page to learn more: <https://buyandsell.gc.ca/procurement-data/contract-history/about-contract-history>. The Procurement Data Dictionary offers a description of each data field contained in the Contract History file, as well as other datasets published on Buyandsell.gc.ca. The XML file is meant to be used as a guide for understanding the various data elements. It is updated on an "as needed" frequency to reflect changes to data elements on Buyandsell.gc.ca.

**Licence:**  
[Open Government Licence - Canada](#)

**Dataset Resources**

Resource Name	Format	Language	Link
Procurement Data Dictionary	XML	Bilingual (English and French)	<a href="#">Download</a>
Complete file: 2009 to today	CSV	Bilingual (English and French)	<a href="#">Download</a>

Have your say  
Rate this dataset  
33 Comment(s)

About this Dataset  
[RSS Feed](#)

Publisher:  
Public Works and Government Services Canada

Tags:  
contracts tenders GETS  
notices supplier  
standing offers  
supply arrangements vendor  
GSIN goods services  
construction

# Do “Something” With Redis



- So, we need to add up all the awarded contract values and associate them with companies (acting as the keys)
- Use a sorted set (scored unique members):
  - Contract Values are the Score
  - Companies Awarded the Contracts are the members of the Set
  - Go through the big list and increment the scores by the contract values for each contract

# Do “Something” With Redis



- In Redis:
  - For Each Contract:
    - ZINCRBY suppliers <contract\_value> <supplier\_name>
  - Okay, so we just need to run through the CSV file and run that command...
- Recommendations for bulk command execution in redis:
  - <http://redis.io/topics/mass-insert>
  - \*sigh\* we have to write some code that generates a file that includes the statements formatted as the underlying redis protocol and pipe that file in (ie: redis has no real bulk load utility)

# Do “Something” With Redis



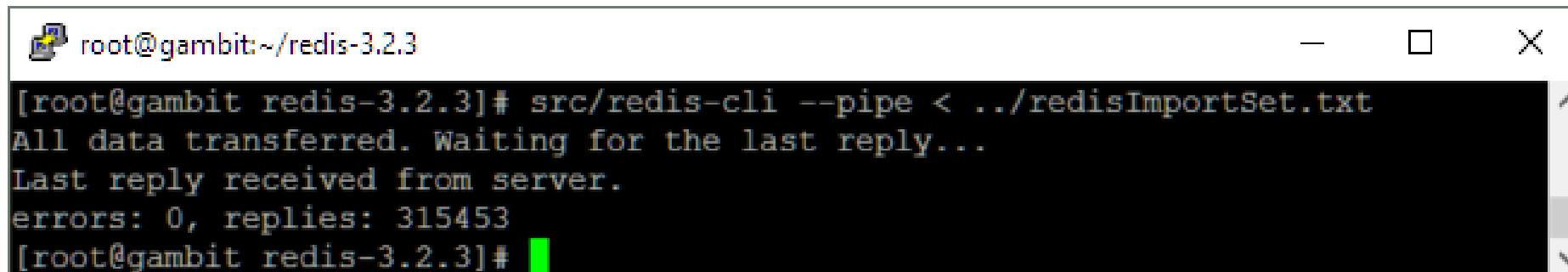
- After the script:
- I omitted the code – it's not that interesting

```
*4
$7
ZINCRBY
$10
suppliers2
$5
61964
$19
ABCO INDUSTRIES LTD
*4
$7
ZINCRBY
$10
suppliers2
$5
25000
$30
BEST WESTERN PLUS ABERCORN INN
```

# Do “Something” With Redis



- Pipe in the Generated File:



A screenshot of a terminal window titled "root@gambit:~/redis-3.2.3". The window contains the following text:

```
[root@gambit redis-3.2.3]# src/redis-cli --pipe < ../redisImportSet.txt
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 315453
[root@gambit redis-3.2.3]#
```

- This is like running the file through netcat (nc) but it happens to count the responses at the same time



# Do “Something” With Redis

- Query the Data – Largest 20 scores: **ZREVRANGE suppliers2 1 20 WITHSCORES**

```
root@gambit:~/redis-3.2.3
127.0.0.1:6379> ZREVRANGE suppliers2 1 20 WITHSCORES
1) "GENERAL DYNAMICS LAND SYSTEMS - CANADA"
2) "5330230307"
3) "GROUPE SIGNATURE SUR LE SAINT-LAURENT"
4) "4572961052"
5) "HALIFAX SHIPYARD"
6) "2976037482"
7) "COMMISSIONAIRES"
8) "2257418353"
9) "GENERAL DYNAMICS ORDNANCE AND TACTICAL SYSTEMS - CANADA INC"
10) "1530425571"
11) "IMPERIAL OIL"
12) "1160561790"
13) "PCL CONSTRUCTORS CANADA INC"
14) "1130049142"
15) "BELL CANADA"
16) "951964515"
17) "MACK DEFENSE, LLC"
18) "836564376"
19) "GLAXOSMITHKLINE INC"
20) "824345297"
21) "TEXTRON SYSTEMS CANADA, INC"
22) "819002940"
23) "VANCOUVER SHIPYARDS CO LTD"
24) "676165933"
25) "BELL HELICOPTER TEXTRON CANADA LTD"
26) "659146472"
27) "BAE SYSTEMS BOFORS AB"
28) "638225158"
29) "IBM CANADA LTD/IBM CANADA LIMITEE"
30) "630961762"
31) "PROJET RESOLVE INC/PROJECT RESOLVE INC"
32) "620865000"
33) "MERCK CANADA INC"
34) "579594188"
35) "LOGISTIK UNICORP INC"
36) "560650442"
```



# Security

Feature	Notes
<b>Authentication</b>	None! (Well... you can password protect with 1 password general access in case your firewalls fail or are misconfigured)
<b>Authorization (access controls)</b>	Zero!
<b>Encryption – Client/Server Communication</b>	Nada! (Well... you can put in an SSL proxy, but that has nothing to do with Redis)
<b>Encryption – Data on Disk</b>	Nope! (Well... you could do it with a separate system, but that has nothing to do with Redis)
<b>Auditing</b>	Zilch!

<http://redis.io/topics/security>

*"Redis is designed to be accessed by trusted clients inside trusted environments. This means that usually it is not a good idea to expose the Redis instance directly to the internet or, in general, to an environment where untrusted clients can directly access the Redis TCP port or UNIX socket. [...] in general, untrusted access to Redis should always be mediated by a layer implementing ACL"*

Here's a question – Should you always trust your administrators? Maybe a pre-requisite for Redis is also jump servers and PIM?

Irony: Salvatore Sanfilippo, the guy who started Redis used to be a security researcher

# Exposed Redis Services with no Authentication



← → C https://slashdot.org/story/16/08/10/237230/linux-trojan-mines-for-cryptocurrency-using-misconfigured-redis-servers

Apps A Java geekA Java Writing a custom Bootstrap 3 + An OpenSSL Certifica 2016 Election For Historical Border Contract History ViacomCM ViacomV10CM

**Slashdot** Stories Firehose > All Popular Polls Deals Submit Search

Topics: Devices Build Entertainment Technology Open Source Science YRO

Catch up on stories from the past week (and beyond) at the [Slashdot story archive](#)

**Linux Trojan Mines For Cryptocurrency Using Misconfigured Redis Servers** (softpedia.com)

Posted by BeauHD on Wednesday August 10, 2016 @09:25PM from the system-to-system dept.

52

An anonymous reader writes:

In another installment of "Linux has malware too," security researchers have discovered a new trojan that [targets Linux servers running Redis](#), where the trojan installs a cryptocurrency miner. The odd fact about this trojan is that it includes a wormable feature that allows it to spread on its own. The trojan, named [Linux.Lady](#), will look for Redis servers that don't have an admin account password, access the database, and then download itself on the new target. The trojan mines for the Monero crypto-currency, the same one used by another worm called [PhotoMiner](#), which targets vulnerable FTP servers. According to a recent Risk Based Security report from last month, there are over [30,000 Redis servers available online](#) without a password, of which [6,000 have already been compromised](#) by various threat actors.

f t in g+

Related Links

← Man Becomes 'Accidental Millionaire' After Jet.com's Sale To Researchers Find Over 6,000 Compromised Redis Installations → Twitter Is Not Legally Responsible For The Rise of ISIS, Rules

database google linux

32

# Exposed Redis Services with no Authentication



https://developers.slashdot.org/story/16/07/09/0448257/researchers-find-over-6000-compromised-redis-installations

Apps A Java geekA Java Writing a custom Bootstrap 3 + An OpenSSL Certific 2016 Election For Historical Border Contract History ViacomCM ViacomV10CM

**Slashdot** Stories Firehose > All Popular Polls Deals Submit

Topics: Devices Build Entertainment Technology Open Source Science YRO

» Slashdot is powered by **your submissions**, so send in your scoop

**Researchers Find Over 6,000 Compromised Redis Installations** (riskbasedsecurity.com)

⚠ Posted by EditorDavid on Saturday July 09, 2016 @10:30AM from the open-source-in-memory-data-structure-store dept.

An anonymous Slashdot reader writes:

Security researchers have discovered over 6,000 compromised installations of Redis, the open source in-memory data structure server, among the tens of thousands of Redis servers indexed by Shodan. "By default, Redis has no authentication or security mechanism enabled, and any security mechanisms must be implemented by the end user."

The researchers also found 106 different Redis versions compromised, suggesting "there are a lot of Redis installations that are not upgrading to the most recent versions to fix any known security issues." 5,892 infections were linked to the same email address, with two more email addresses that were both linked to more than 200. "The key take away from this research for us has been that insecure default installations continue to be a significant issue, even in 2016."

Redis "is designed to be accessed by trusted clients inside trusted environments," according to its documentation. "This means that usually it is not a good idea to expose the Redis instance directly to the internet or, in general, to an environment where untrusted clients can directly access the Redis TCP port or UNIX socket... Redis is not optimized for maximum security but for maximum performance and simplicity."

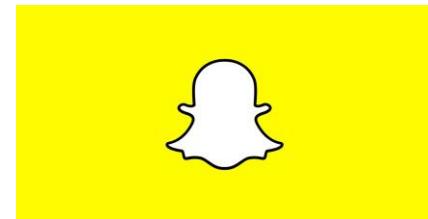
# Where it's used



Good List: <http://redis.io/topics/whos-using-redis>

Mainly “Web Scale” kinds of use cases

craigslist



# Potential Drawbacks



- It is clearly built for performance above all else, so certain functions suffer:
  - Security
  - Utilities
  - Extended functions
- Not exactly user friendly
  - Bulk importing data required code and knowing the protocol
  - Something simple like scanning through keys implemented with performance over usability in mind
- Might not be appropriate for analytics (but analytic systems are often not appropriate for transactions – choose it for the right use cases)

# Similar Systems



- Memcached is often cited as an alternative for pure caching use cases (key-object store in memory)
- Apache Ignite is an in-memory key-value store, but relies on an external database (nosql/relational) for persistence
- Apache Cassandra (coming up next) is a hybrid key-value/column-oriented data store, so the behavior is somewhat analogous but feels very different





# General Architecture

- Cassandra is a hybrid key-value/tabular data store
- Instead of master/slaves, replication occurs across the cluster instead of having dedicated slaves
- Nodes are organized in a ring
- CAP: Cassandra values availability and partition tolerance (it can be configured to be more consistent at the cost of increased latency (ie: availability) – but you can't get row locking)
- Something that takes some getting used to: In Cassandra, you build all your tables with data retrieval as the primary concern – multiple tables/copies of data for each kind of SELECT you plan to do is common
- You interact with Cassandra using a SQL flavored language called CQL



# Origins

- Cassandra comes from Facebook – Created to power inbox search features  
The Facebook logo, which is a dark blue rounded rectangle containing the word "facebook" in white, lowercase, sans-serif letters.
- One of the lead developers came from Amazon where he helped author Amazon DynamoDB (key-value store, ring architecture, other things)  
The Amazon DynamoDB logo, which features three yellow cubes arranged in a 2x2 grid (top-left cube is yellow, top-right is orange, bottom-left is orange, bottom-right is yellow), followed by the text "amazon" in a small, black, sans-serif font and "DynamoDB" in a larger, bold, black, sans-serif font.
- Also took some inspiration from Google's BigTable Paper (column families, SSTables , other things)  
The Google logo, which is the word "Google" in its signature multi-colored, sans-serif font.
- “Cassandra” probably chosen on purpose – cursed Oracle in Greek Mythology who can see the future, but who’s curse is no one believes her



# Do “Something” With Cassandra

- Elections Ontario keeps a database of Ontario political contributions people can query
  - <https://www3.elections.on.ca/internetapp/efsearch.aspx>

The screenshot shows the 'Election Finances' section of the Elections Ontario website. At the top, there are links for 'Voting in Ontario', 'Political Entities', 'Resources', 'Media Centre', and 'About Us'. Below this is a search interface with a 'Financial Search' field containing the radio button option 'Search by Contributors'.

- Question: Across all “events” (leadership races, individual contributions, yearly contributions, general election contributions), who were the largest contributors to each party - 2014 + 2015



# Do “Something” With Cassandra

- In Cassandra, while you define tables, ***they are not relational***. Things you can't do:
  - Joins (instead denormalize and use collections)
  - No GROUP BY statements (but very basic aggregations are recently supported)
  - Sorting has to be defined during data storage – more on that later
- Create a User Defined Type (UDT) for individual contribution details – we'll use these in a collection

```
root@gambit:~/apache-cassandra-3.7/bin
cqlsh:mykeyspace> CREATE TYPE IF NOT EXISTS mykeyspace.contribution_item(contributed_to t
ext, affiliation text, year int, event text, type text, amount int);
cqlsh:mykeyspace>
```



# Do “Something” With Cassandra

- Create the table (attempt 1). I think to myself “keys are normally sorted and partitioning based on amount contributed would create a pretty good distribution across nodes:

```
create table contributors(  
    amnt int,  
    party text,  
    contibutor text,  
    contribs set<frozen<contribution_item>>,  
    PRIMARY KEY (amnt, party, contibutor)  
)
```

- The first column in the primary key is the partition key



# Do “Something” With Cassandra

- No Dice – Cassandra does not want to retrieve all the data from a bunch of nodes and sort them (makes sense actually):

```
root@gambit:~/apache-cassandra-3.7/bin
cqlsh:mykeyspace> create table contributors_amnt(
    ... amnt int,
    ... party text,
    ... contibutor text,
    ... contribs set<frozen<contribution_item>>,
    ... PRIMARY KEY (amnt, party, contibutor)
    ... ) ;
cqlsh:mykeyspace> select * from contributors_amnt ORDER by amnt;
InvalidRequest: code=2200 [Invalid query] message="ORDER BY is only supported when the partition key is restricted by an EQ or an IN."
cqlsh:mykeyspace>
```

- So, I can sort, but I have to include the partition key in the query



# Do “Something” With Cassandra

- Second Attempt: Create the table and define the clustering to force sorting on storage of the data around amnt:

```
PUTTY (inactive)
cqlsh:mykeyspace> create table contributors(
...   party text,
...   amnt int,
...   contibutor text,
...   contribs set<frozen<contribution_item>>,
...   PRIMARY KEY (party, amnt, contibutor)
... ) WITH CLUSTERING ORDER BY(amnt DESC);
```

- Partitioning on party. Not perfect distribution either but at least I only have to read one partition per query (which is something Cassandra encourages)



# Do “Something” With Cassandra

- I have a table now, so take the Election Ontario Flat Files:

C:\Users\john.haldeman\Desktop\ONPolicalContributions.txt - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

ONPolicalContributions.txt

```
1 Name,Contributed To,Political Affiliation,Year,Event,Type,Amount
2 1003694 ONTARIO INC (CAREPARTNERS) ,Ontario Liberal Party,LIB,2014,2014 Annual Period,Individual,1910
3 1026517 ONTARIO LTD ,Ontario Liberal Party,LIB,2014,2014 Annual Period,Individual,690
4 1035587 ONTARIO INC ,Progressive Conservative Party of Ontario,PCP,2014,2014 Annual Period,Individual,950
5 1037416 ONTARIO INC ,Ontario Liberal Party,LIB,2014,2014 Annual Period,Individual,468
6 1039591 ONTARIO LTD O/A D & D ENTERPRISES ,Progressive Conservative Party of Ontario,PCP,2014,2014 Annual Period,Ind
7 1043 BLOOR INC ,Progressive Conservative Party of Ontario,PCP,2014,2014 Annual Period,Individual,379.52
```



# Do “Something” With Cassandra

- Unflatten them to create collections and sum the contributions (I left out of code for this one):

C:\Users\john.haldeman\workspace\RedisGen\cassandraImportSet.txt - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

ONPolicalContributions.txt cassandraImportSet.txt

```
1 "PCP",1000,"GEORGE STEIN LTD","{{contributed_to: 'John Yakabuski', affiliation: 'PCP', year: 2014, event: 'General Election 2014', type: 'Individual', amount: 1000}}"
2 "PCP",210,"886110 ONTARIO LTD","{{contributed_to: 'Patrick Brown', affiliation: 'PCP', year: 2014, event: '2015 Progressive Conservative Party of Ontario Leadership Contes
3 "PCP",140,"SHARPE ANNE","{{contributed_to: 'Progressive Conservative Party of Ontario', affiliation: 'PCP', year: 2015, event: '2015 Annual Period', type: 'Individual', am
4 "LIB",600,"DONSON LYNDA","{{contributed_to: 'Ontario Liberal Party', affiliation: 'LIB', year: 2014, event: '2014 Annual Period', type: 'Individual', amount: 300},{contrib
5 "PCP",300,"MILES MARGARET","{{contributed_to: 'Progressive Conservative Party of Ontario', affiliation: 'PCP', year: 2015, event: '2015 By-Election ED088', type: 'Individu
6 "PCP",200,"MACKIE ALAN","{{contributed_to: 'Progressive Conservative Party of Ontario', affiliation: 'PCP', year: 2014, event: 'General Election 2014', type: 'Individual',
7 "LIB",200,"STEWART JOHN","{{contributed_to: 'Ontario Liberal Party', affiliation: 'LIB', year: 2014, event: 'General Election 2014', type: 'Individual', amount: 200}}"
8 "LIB",350,"MCGRATH JOHN","{{contributed_to: 'Ontario Liberal Party', affiliation: 'LIB', year: 2015, event: '2015 Annual Period', type: 'Individual', amount: 350}}"
```

- Basically I did the GROUP BY outside of Cassandra, but Cassandra experts might be able to do this all in-engine? Maybe with Spark?  
(more on Spark later)



# Do “Something” With Cassandra

- Import the data:

```
root@gambit:~/apache-cassandra-3.7/bin
cqlsh:mykeyspace> COPY contributors from 'cassandraImportSet.txt';
Using 1 child processes

Starting copy of mykeyspace.contributors with columns [party, amnt, contibutor,
contribs].
Processed: 46566 rows; Rate: 326 rows/s; Avg. rate: 684 rows/s
46566 rows imported from 1 files in 1 minute and 8.075 seconds (0 skipped).
cqlsh:mykeyspace>
```

- (For larger data sets, they recommend you compile the data directly into SSTables instead using the Cassandra Java API)

# Do “Something” With Cassandra



```
root@gambit:~/apache-cassandra-3.7/bin
cqlsh:mykeyspace> select party, amnt, contibutor from contributors where party='LIB' LIMIT 20;

  party | amnt   | contibutor
-----+-----+
    LIB | 60630 | LAZARIDIS MICHAEL
    LIB | 60548 | ROGERS COMMUNICATIONS INC
    LIB | 59925 | INSURANCE BUREAU OF CANADA IBC
    LIB | 59005 | TONG OPHELIA
    LIB | 57432 | CARPENTERS DISTRICT COUNCIL OF ONTARIO
    LIB | 55749 | UNION GAS LTD
    LIB | 54060 | TERANET INC
    LIB | 53630 | GREENFIELD SPECIALTY ALCOHOLS INC
    LIB | 52363 | LIFELABS MEDICAL LABORATORY SERVICES INC
    LIB | 51310 | UNIFOR (UNION FOR CANADA)
    LIB | 51240 | SHOPPERS DRUG MART INC
    LIB | 50466 | RENTECH WP CANADA ULC
    LIB | 50372 | IRON WORKERS DISTRICT COUNCIL OF ONTARIO
    LIB | 49845 | SAMSUNG RENEWABLE ENERGY INC
    LIB | 48990 | BANK OF NOVA SCOTIA (SCOTIABANK)
    LIB | 48512 | APOTEX INC
    LIB | 47670 | ONTARIO COUNCIL OF PAINTERS
    LIB | 47368 | ONTARIO HOME BUILDERS ASSOC INC OHBA
    LIB | 46398 | LOC 793 INTERNATIONAL UNION OF OPERATING ENGINEERS
    LIB | 46321 | ONTARIO LONG TERM CARE ASSOC OLTCA

(20 rows)
cqlsh:mykeyspace>
```



# Do “Something” With Cassandra

```
root@gambit:~/apache-cassandra-3.7/bin
cqlsh:mykeyspace> select party, amnt, contibutor from contributors where party='NDP' LIMIT 20;

   party | amnt | contibutor
-----+-----+
    NDP |  69825 | DIST 6 USW
    NDP |  66290 | AMALGAMATED TRANSIT UNION ONT
    NDP |  64627 | LOC 113 AMALGAMATED TRANSIT UNION
    NDP |  57439 | LOC 1 SEIU
    NDP |  54914 | OPSEU
    NDP |  52661 | LOC 353 IBEW
    NDP |  43955 | THE SOCIETY OF ENERGY PROF
    NDP |  41387 | OECTA ONTARIO ENGLISH CATHOLIC TEACHERS ASSOC
    NDP |  40409 | USW POLITICAL ACTION COMMITTEE
    NDP |  37996 | LOC 1998 USW
    NDP |  35627 | LOC 46 UA PLUMBING & PIPE FITTING
    NDP |  35345 | BRUCE POWER INC
    NDP |  34580 | CANADIAN LABOUR CONGRESS
    NDP |  34550 | LOC 1000-A UFCW
    NDP |  34322 | UNIFOR THE UNION
    NDP |  32575 | ONTARIO REAL ESTATE ASSOC
    NDP |  31868 | LIFELABS INC
    NDP |  31385 | ETFO ELEMENTARY TEACHERS FED OF ONT
    NDP |  31322 | ONTARIO CUPE
    NDP |  30290 | PROVINCIAL BUILDING & CONSTRUCTION TRADES COUNCIL ONTARIO

(20 rows)
cqlsh:mykeyspace>
```

# Do “Something” With Cassandra



```
root@gambit:~/apache-cassandra-3.7/bin
cqlsh:mykeyspace> select party, amnt, contibutor from contributors where party='PCP' LIMIT 20;

  party | amnt      | contibutor
-----+-----+
    PCP | 133250    |
    PCP | 127830    | FEDERATION OF RENTAL HOUSING PROVIDERS OF ONTARIO
    PCP | 110271    | INSURANCE BUREAU OF CANADA
    PCP | 74578     | ONTARIO REAL ESTATE ASSOC
    PCP | 60230     | MILBORNE REAL ESTATE INC
    PCP | 49680     | ELEMENT FINANCIAL CORP
    PCP | 47679     | MEDIPAC INTERNATIONAL INC
    PCP | 45425     | ROTO-MILL INC
    PCP | 44765     | CANADIAN TIRE CORP LTD
    PCP | 44103     | 1714486 ONTARIO LTD
    PCP | 43975     | ORLANDO CORP
    PCP | 43250     | JETPORT INC
    PCP | 43250     | STRONACH CONSULTING CORP
    PCP | 42389     | BARRICK GOLD CORP
    PCP | 39624     | UNION GAS LTD
    PCP | 39261     | MOLSON COORS CANADA INC
    PCP | 38465     | ROGERS COMMUNICATIONS INC
    PCP | 37915     | LIFELABS INC
    PCP | 37712     | 611428 ONTARIO LTD
    PCP | 35000     | MUNK PETER

(20 rows)
cqlsh:mykeyspace>
```

# Do “Something” With Cassandra



```
root@gambit:~/apache-cassandra-3.7/bin
cqlsh:mykeyspace> select party, amnt, contibutor from contributors where party='GPO' LIMIT 20;

   party |   amnt |   contibutor
-----+-----+
    GPO | 12100 |
    GPO |  6985 |
    GPO |  6905 |
    GPO |  6310 |
    GPO |  6284 |
    GPO |  6030 |
    GPO |  6000 |
    GPO |  5870 |
    GPO |  5810 |
    GPO |  5591 |
    GPO |  5530 |
    GPO |  5450 |
    GPO |  5360 |
    GPO |  5000 |
    GPO |  4630 |
    GPO |  4625 |
    GPO |  4535 |
    GPO |  4466 |
    GPO |  4400 |
    GPO |  4152 | 1912206 ONTARIO INC O/A ELLMAN DESIGN

   VETTESE SHAROLYN MATHIEU
   KNIGHT GEOFFREY
   SCHREINER MIKE
   EASTO HAYLEY
   ELLIOTT ROBERT S
   MANICKAM NARA
   ROCHON CLAUDETTE
   WATSON AMY
   DAYE MARK
   HARTWELL GEORGE
   1912206 ONTARIO INC O/A ELLMAN DESIGN

(20 rows)
cqlsh:mykeyspace>
```

# Security



Feature	Notes
<b>Authentication</b>	Native Password Authentication is available. You could extend the authenticator classes to use whatever you like. The Enterprise distributions (eg: Datastax) have authenticators for things like LDAP, Kerberos.
<b>Authorization (access controls)</b>	Native Role Based Authorization is available. You could extend the authorization classes to use whatever you like.
<b>Encryption – Client/Server Communication</b>	Yes: TLS/SSL available
<b>Encryption – Data on Disk</b>	Enterprise Distros (eg: Datastax) have TDE implementations with KMIP integrations. Or you can use something else.
<b>Auditing</b>	None, except if you use an Enterprise distribution (eg: Datastax) or if you use a third party auditing framework (eg: Guardium)



# Where It's Used

- Many of the most famous examples chose Cassandra for maximum control and for writing a lot of data quickly (with reads being relatively infrequent)
- More tolerant to amounts of data larger than the available memory (where MongoDB and Redis historically don't do well)



- Instagram migrated from Redis (not from a relational DBMS) to Cassandra for certain functions (the Facebook influence maybe helped?)



# Potential Drawbacks

- Cassandra Data Modeling Feels “Unusual”
  - <http://www.datastax.com/dev/blog/basic-rules-of-cassandra-data-modeling>

*“Try to determine exactly what queries you need to support. [...] Changes to just one of these query requirements will frequently warrant a data model change for maximum efficiency. [...] To put this another way, each table should pre-build the “answer” to a high-level query that you need to support. If you need different types of answers, you usually need different tables. This is how you optimize for reads.”*
  - It kind of seems like a pain that in order to get another view of the data, you have to rewrite the data into a differently organized table



# Potential Drawbacks

- More than some of the other DBMSs explored here, Cassandra does sometimes feel like a moving target (constantly changing). It can be difficult to know if what you are reading was a consideration in the past or is one now
- The official documentation that's supposed to set you straight apparently an evolving work in progress itself:

The screenshot shows a web browser window with the following details:

- Title Bar:** Documentation
- Address Bar:** cassandra.apache.org/doc/latest/architecture/guarantees.html
- Toolbar:** Back, Forward, Stop, Refresh, Home, Search, Favorites, RSS feed icon (179).
- Page Headers:** Apache Software Foundation, Apache Cassandra, Documentation, Architecture, Guarantees.
- Page Content:**
  - Logo:** Apache CASSANDRA™ with eye icon.
  - Search:** Search docs button.
  - Navigation:** Home, Download, Documentation, Community.
  - Section:** Guarantees
  - Content:** Todo section containing the text "todo".

# Similar Systems



- The most similar system in that they both have closely tied origins to Google's BigTable is HBASE:



- HBASE is built on Hadoop which comes next

# General Architecture



- HDFS
  - The Hadoop Distributed Filesystem – A single filesystem across many nodes
  - Provides the data for everything that runs on Hadoop
  - Two Major Components:
    - NameNodes: Provide metadata services (but no actual data flows through them). They also orchestrate replication
    - DataNodes: Do the actual storage and data management
- More on HDFS: <https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>

# General Architecture

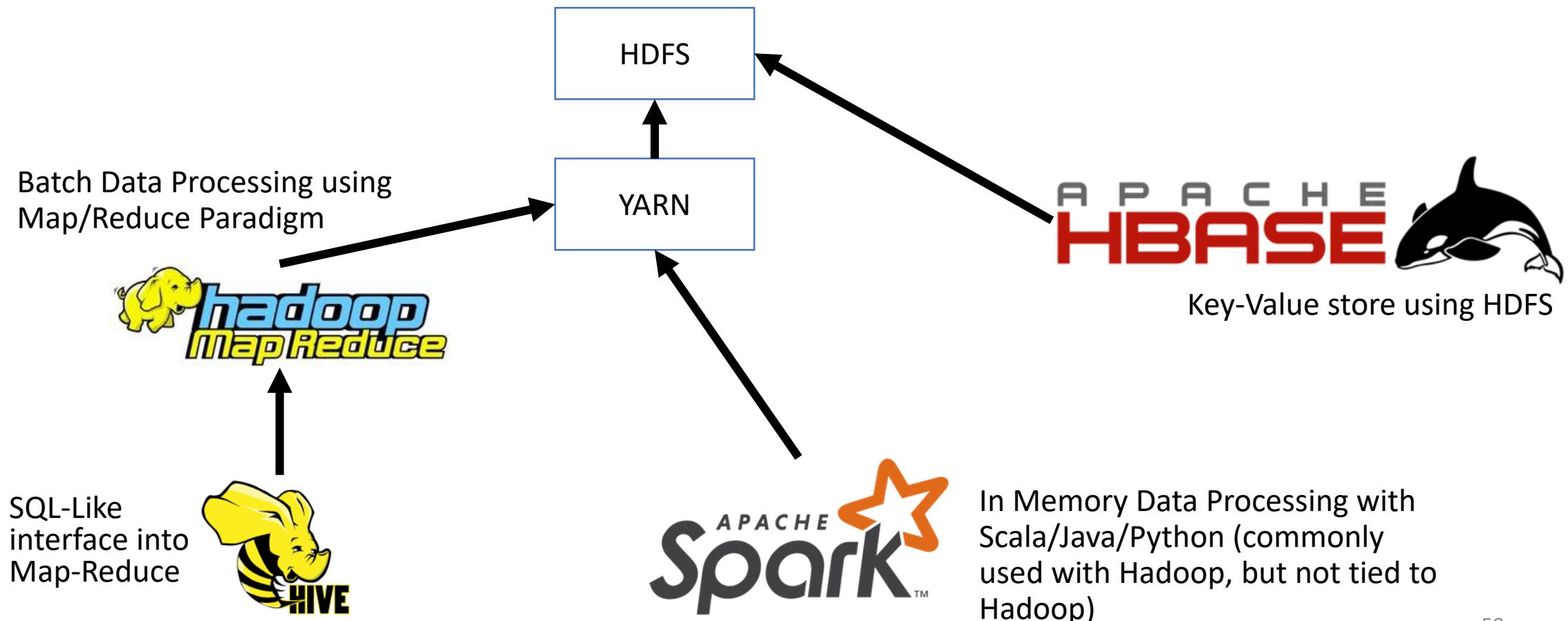


- YARN
  - Yet Another Resource Negotiator
  - Major component is the ResourceManager: Allocates and manages cluster resources (a system scheduler)
- More on YARN: <http://hortonworks.com/blog/apache-hadoop-yarn-concepts-and-applications/>

# General Architecture



- “Stuff” built on top of it (Nowhere near a complete list)



# Origins



Google



The Google Filesystem and MapReduce Papers

(<http://research.google.com/archive/gfs.html>    <http://research.google.com/archive/mapreduce.html> )



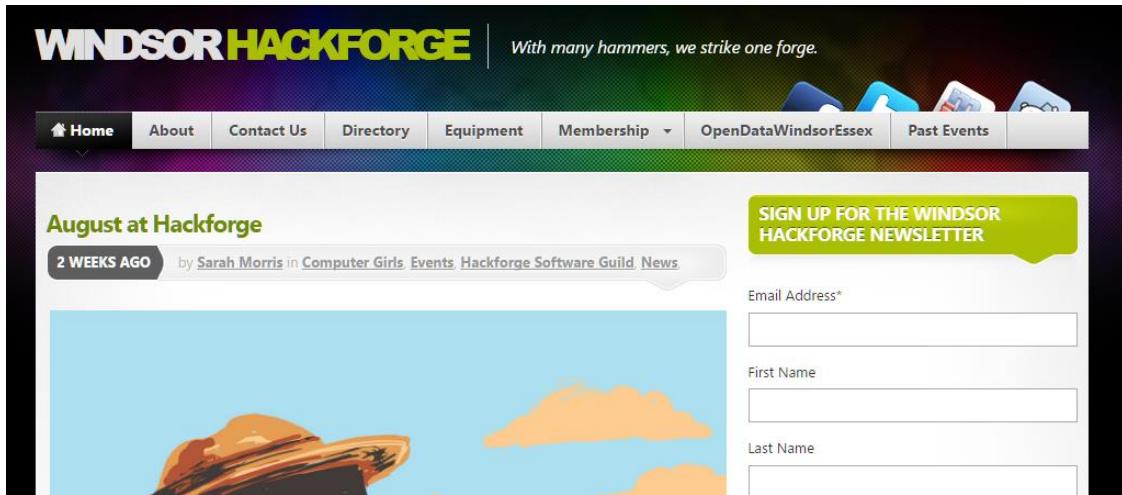
YAHOO!



# Do “Something” With Hadoop



- Hey, websites are data:



- Let's Analyze Hackforge's website with Hadoop and Spark using word and link counts – what words appear most often for:
  - <http://hackf.org/>

# Do “Something” With Hadoop



- Get the data with wget recursive (not in Hadoop yet):

```
wget --limit-rate=200k --no-clobber --convert-links --random-wait -r -p -E -e robots=off -U mozilla  
http://hackf.org/
```

- Next, convert the site to pure text with a HTML parser. A Lynx dump works for something quick and dirty (not in Hadoop yet):

```
lynx -nolist -dump justhtml/_2013_06_01_hackforge-grand-opening_index.html > justtext/_2013_06_01_hackforge-grand-opening_index.txt
```

```
lynx -nolist -dump justhtml/_2013_07_06_forum-is-live_index.html > justtext/_2013_07_06_forum-is-live_index.txt
```

```
lynx -nolist -dump justhtml/_2013_07_11_what-is-an-oscilloscope_feed_index.html > justtext/_2013_07_11_what-is-an-  
oscilloscope_feed_index.txt
```

```
lynx -nolist -dump justhtml/_2013_07_11_what-is-an-oscilloscope_index.html > justtext/_2013_07_11_what-is-an-oscilloscope_index.txt
```

....

# Do “Something” With Hadoop



- Copy the data from the local file system to HDFS (Now we are in Hadoop):
  - `hadoop fs -copyFromLocal *.txt /data/hackforg/text`

```
root@sandbox:~# hadoop fs -ls /data/hackforg/text | head
Found 429 items
-rw-r--r--  3 root hdfs      4232 2016-08-06 02:12 /data/hackforg/text/2013_06
_01_hackforge-grand-opening_index.txt
-rw-r--r--  3 root hdfs      1754 2016-08-06 02:12 /data/hackforg/text/2013_07
_06_forum-is-live_index.txt
-rw-r--r--  3 root hdfs     3733 2016-08-06 02:12 /data/hackforg/text/2013_07
_11_what-is-an-oscilloscope_feed_index.txt
-rw-r--r--  3 root hdfs    11430 2016-08-06 02:12 /data/hackforg/text/2013_07
_11_what-is-an-oscilloscope_index.txt
-rw-r--r--  3 root hdfs    11430 2016-08-06 02:12 /data/hackforg/text/2013_07
_11_what-is-an-oscilloscope_trackback_index.txt
-rw-r--r--  3 root hdfs    3588 2016-08-06 02:12 /data/hackforg/text/2013_07
_12_we-are-now-adafruit-dealers_feed_index.txt
-rw-r--r--  3 root hdfs    4618 2016-08-06 02:12 /data/hackforg/text/2013_07
_12_we-are-now-adafruit-dealers_index.txt
-rw-r--r--  3 root hdfs    4618 2016-08-06 02:12 /data/hackforg/text/2013_07
_12_we-are-now-adafruit-dealers_trackback_index.txt
-rw-r--r--  3 root hdfs     174 2016-08-06 02:12 /data/hackforg/text/2013_07
_15_parts-database-in-beta_feed_index.txt
[root@sandbox:~]#
```

# Do “Something” With Hadoop



- Use Spark with some Scala to count the words:

The image shows a screenshot of the Zeppelin notebook interface. At the top, there's a blue header bar with the Zeppelin logo (a white airship icon), the word "Zeppelin", and navigation links for "Notebook", "Interpreter", and "Configuration". Below the header, the main workspace has a title "Simple Sorted Word Count" and a toolbar with various icons for running, saving, and deleting cells. The central area contains a Scala code cell with the following content:

```
val textFile = sc.textFile("hdfs:///data/hackforg/text")
val counts = textFile.flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey(_ + _)
  .map(item => item.swap)
  .sortByKey(false)
counts.saveAsTextFile("hdfs:///data/hackforg/text_out14")
```

Below the code cell, the interpreter output is shown:

```
textFile: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <console>:30
counts: org.apache.spark.rdd.RDD[(Int, String)] = ShuffledRDD[8] at sortByKey at <console>:35
```

Took 102 seconds

# Do “Something” With Hadoop



- HDFS shell command to view the results file:



```
%sh hadoop fs -cat /data/hackforg/text_out14/part*
```

```
(205711,)  
(7736,the)  
(7506,to)  
(7492,and)  
(5879,*)  
(4726,of)  
(4429,a)  
(4256,for)  
(4047,Hackforge)  
(3523,+)  
(3304,in)  
(2992,Windsor)  
(2949,is)  
(2920,by)  
(2753,on)  
(2245,2016)  
(2173,with)  
(2113,/>)
```

# Do “Something” With Hadoop



- HDFS shell command to view the results file:

## \* Simple Sorted Word Count [

```
(791,as)
(708,Sarah)
(695,ago)
(688,our)
(688,Feed)
(669,May)
(658,Ontario)
(657,Morris)
(641,have)
(641,as)
(639,one)
(637,new)
(621,July)
(606,Event)
(599,Trillium)
(599,Open)
(588,us)
(584,August)
(579,women)
(561,Computer)
(557,about)
(540,Data)
```

# Do “Something” With Hadoop



- Scala on Spark to look at link counts in HTML:

```
* Link Count  ⏪ ⏴ ⏵ ⏷ ⏸ ⏹ ⏺ ⏻ ⏳ ⏴
```

```
val textFile = sc.textFile("hdfs:///data/hackforg/html")  
  
val counts = textFile.filter(line => line.contains("href"))  
    .flatMap(hrefline => hrefline.split(" "))  
    .filter(word => word.contains("href"))  
    .map(word => word.replace("href=", ""))  
    .map(word => word.split(">")(0))  
    .map(linkWord => (linkWord, 1))  
    .reduceByKey((count1, count2) => (count1 + count2))  
    .map(item => item.swap)  
    .sortByKey(false)  
  
counts.saveAsTextFile("hdfs:///data/hackforg/html_out7")  
  
textFile: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[16] at textFile at <console>:30  
counts: org.apache.spark.rdd.RDD[(Int, String)] = ShuffledRDD[27] at sortByKey at <console>:40  
Took 19 seconds
```

# Do “Something” With Hadoop



- Results:

```
%sh hadoop fs -cat /data/hackforg/html_out7/part*
```

```
(505,"http://hackf.org/feed/")
(504,"http://hackf.org/")
(483,"http://hackf.org/author/sarah/")
(320,"http://hackf.org/category/events/")
(275,"https://twitter.com/hackforge")
(274,"http://hackf.org/category/news/")
(274,"http://hackf.org/opendatawindsorsex/")
(264,"http://hackf.org/join-us/")
(257,"http://hackf.org/join-us/code-of-conduct/")
(257,"http://hackf.org/join-us/rules/")
(256,"http://hackf.org/member-directory/")
(253,"http://hackf.org/contact-us/")
(253,"http://hackf.org/comments/feed/")
(252,"https://www.facebook.com/Hackforge")
(252,"#page")
(252,'http://hackf.org/wp-content/plugins/mailchimp//css/flick/flick.css?ver=3.9.13')
(252,"http://webchat.freenode.net/?channels=#hackforge")
(252,"http://hackf.org/2016/06/")
(252,"http://hackf.org/2016/02/")
(252,"http://digitalnature.eu/")
(252,"http://wordpress.org/")
(252,"/favicon.ico")
(252,'http://hackf.org/?mcsf_action=main_css&ver=3.9.13')
(252,"http://hackf.org/2015/09/")
(252,'http://hackf.org/wp-content/themes/mystique-child/simple-staff-list-custom.css?ver=3.9.13')
(252,"http://hackf.org/about/")
(252,"http://www.reddit.com/r/hackforge/search?q=hackforge&sort=relevance&restrict_sr=on&t=all")
(252,"http://hackf.org/2016/07/")
```

# Security



Feature	Notes
<b>Authentication</b>	Kerberos and Nothing
<b>Authorization (access controls)</b>	At HDFS there are similar access controls to regular filesystems. For the systems on top (HBASE, HIVE, etc.), it varies. Since each system has their own access control mechanisms and structures, it can be a complex task to design entitlements. There are systems to try help manage the complexity (eg: Apache Ranger)
<b>Encryption – Client/Server Communication</b>	HDFS Transparent encryption handles end to end encryption (at rest and in the communication stream). That being said, that just encrypts the data with the services that interact with HDFS. Most services on top of HDFS (like HBASE, HIVE, Spark, Impala etc.) will have varying support/configurations for communication encryption.
<b>Auditing</b>	Various native audit logging at the various layers is available

# Where It's Used



- Becoming the standard in big data analytics – quite common. Here are some examples outside of web technology:



# Potential Drawbacks



- It is a complex technology – a framework for data processing rather than a DBMS with a single interface
- HBASE excepted, most technologies that run on HDFS are primarily for batch processing of OLAP workloads – so use it for the right use cases. There aren't a lot of options for mixed workloads (but when you are dealing with Big Data, there may be no options for mixed workloads anyway)
- Some people struggle to understand the Map-Reduce paradigm, which is the traditional one used on Hadoop

# Similar Systems



- There seem to be similar systems for each component in the ecosystem, but it's difficult to find something that is similar to Hadoop as a whole

# NoSQL Cocktail Hour



I'm good at helping you with ***almost*** everything and I am easy to talk to (but arguably postgresql is both those things as well – but I am schema-less)

If you have a problem that involves lots of data, there is probably something somewhere in the Hadoop ecosystem that will help you out



I am extremely fast as long as I can keep it all in memory



I am good at writing things down quickly and telling you things quickly. It just gets awkward if the views you want change often or if you need me to summarize the data