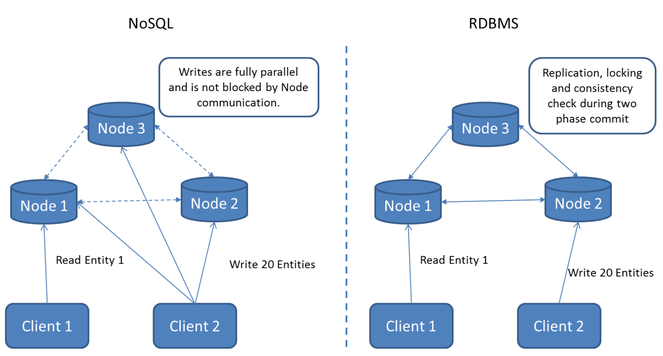
Applications of Open-source NoSQL Database Systems for Astronomical Spatial and Temporal Data

Min-Su Shin (Korea Astronomy and Space Science Institute)

1. Introduction

What's NoSQL? Most astronomers are familiar with the SQL which have rows and columns in pre-defined fixed formats. NoSQL stores the data in many different formats such as key – value pairs (in Redis and Hbase/OpenTSDB).

Why NoSQL? While most SQL database systems do not scale well, many NoSQL solutions work well as described in the following figure.



https://www.dynatrace.com/blog/nosql-or-rdbms-are-we-asking-the-right-questions/

SQL VS. NOSQL OVERSIMPLIFIED

SELECT * FROM Customers_tbl WHERE Last_Name='Smith';

Cust_No	Last_Name	First_Name
560779	Smith	Juan
207228	Smith	George
173996	Smith	Ben
477610	Smith	Conrad

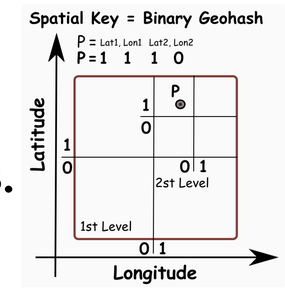
Get customer.firstname,customer.lastname,customer.productID.* where Last_Name='Whitelock'

Key	Value
746133	Firstname: George Lastname: Whitelock productID: 2012: 5
135225	Firstname: Luke Lastname: Whitelock productID: 1285: 1 1077: 5
884256	Firstname: Sam Lastname: Whitelock productID: 1442: 2

https://arstechnica.com/information-technology/2016/03/to-sql-or-nosql-thats-the-database-question/

2. Redis: NoSQL in-memory platform for astronomical spatial data

Goal We test whether Redis can be used easily as a useful platform of data analysis in astronomy, particularly focusing on astronomical spatial data and GEOHASH (i.e. Z-order space-filling curve) in Redis.

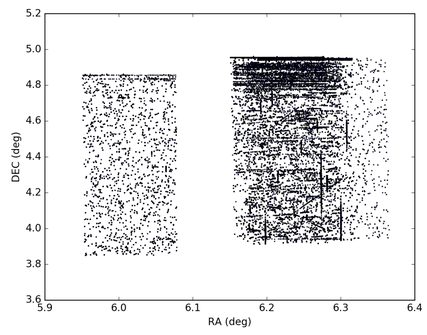


https://karussell.wordpress.com/2012/05/23/spatial-keys-memory-efficient-geohashes/

Possible applications
- Implementation of fast astrometric calibration in memory.
- Quick neighbor search and clustering with given positions and search radii.

Experiments a) modifying the source codes of Redis to support RA/DEC coordinates, b) ingesting a small dataset of KMTNet DEEP-South time-series observations as Redis GEOHASH objects, c) conducting many neighbor searches, and d) associating closely located measurements as a single cluster of measurements (i.e., constructing light curves of objects). The dataset includes about 13.2 million photometric measurements.

Results
- Redis API is already well designed, and we need only minor modifications of Redis for the RA range instead of the longitude range.
- The following example Python code is used to ingest the 13.2 million measurements within 33 minutes (wall-clock time) in case of using a single thread in a server-class machine.
- About 220 microseconds for neighbor search with Redis GEORADIUS and 4 arcseconds radius.
- About 12.8 measurements (i.e. 96.8%) are found in light curves with more than 30 measurements.



Example distribution of (RA, DEC) in a single amplifier section.

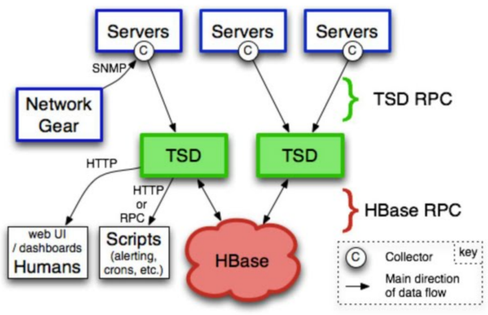
```
in_reader = csv.reader(use_csvfile, delimiter = ' ')
for row in in_reader:
    # example: geopos pos "amp1/kmtc.20150818.025457_1.frame:34"
     db_member = db_member_prefix + ':' + row[0]
     # example: hgetall "amp1/kmtc.20150818.025457_1.frame:34"
     db_phot_key = db_member
     db_phot_value = {'RA' : row[3], 'DEC' : row[4], ₩
              'MJD' : row[5], 'MAG' : row[6], 'MAGERR' : row[7]}
     db_pos_RA = float(row[3])
     db_pos_DEC = float(row[4])
     redis_conn.geoadd(use_pos_key, db_pos_RA, db_pos_DEC, ₩
               db_member)
     redis_conn.hmset(db_phot_key, db_phot_value)
```

3. OpenTSDB: NoSQL database system for astronomical temporal data

Goal We adopt OpenTSDB as a special large-scale time-series database system for future big time-series data such as LSST level 1 alerts. The expected alert rate is about 1 million per hour.

Possible applications
- Alert database of VOEvent streams from future time-domain surveys such as ZTF survey.

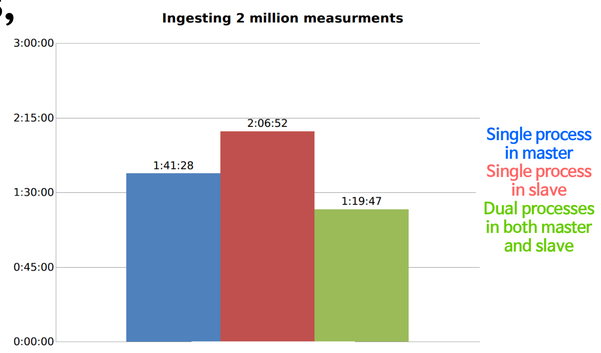
Experiments a) ingesting a small test dataset of time-series observations acquired in VVV survey (see the example Python code below), b) conducting many time-range searches, and c) conducting several object-based searches.



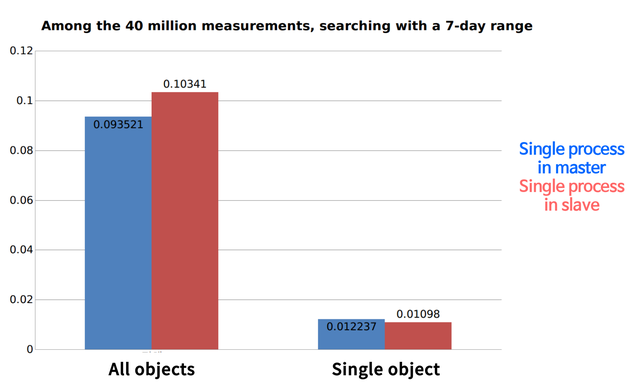
OpenTSDB is a distributed, scalable Time Series Database (TSDB) written on top of HBase. OpenTSDB was written to store, index and serve metrics collected from computer systems (network gear, operating systems, applications) at a large scale, and make this data easily accessible and graphable.

Results In the test environment composed of two low-cost old Pentium machines,

Time to ingest the data



Time to search the data (secs)



- OpenTSDB is used successfully to manage a reasonable amount of time-series data even in a poor computation environment like the test environment.

4. Plan

- Experiments to improve performance in both Redis and OpenTSDB systems.
- Finding objects in time-series data as asteroid or transient candidates by exploiting the fast data accessibility in the NoSQL environments.

Data ingestion

```
url = http://192.168.0.1:4242/api/put
data = {
      "metric": "use_ksmag",
      "timestamp": utc,
      "value": use_ksmag
       tags : {
              "use_iauname": use_iauname
               }
         }
ret = request.post(url, data=json.dump(data))
```

Time-range search

```
url = "http://192.168.0.1:4242/api/query
          ?start=2000/01/01-00:00:00
          &end=2017/08/24-23:59:59
          &m=none:use_ksmag"
ret = requests.get(URL)
```