Amateur Radio Propagation Analysis with Erlang/OTP and Riak

Kenji Rikitake, JJ1BDX and N6BDX
Basho Japan KK / Basho Technologies, Inc.
21-MAR-2013
Executive summary

Sensor networks have become an emerging aspect of amateur radio: what you transmit can be reported from the other end of the world, and will be a part of "big data" archives (you have been warned)

A newbie Riak engineer (me) set up Riak with Yokozuna on FreeBSD, and made a prototype to analyze two of those archives, and finally had happy results after a week of struggling

Conclusion: Riak with Yokozuna fits very well for data analysis, empowered by Apache Lucene/Solr
Amateur (ham) radio

US 47 CFR (aka FCC Rules) §97.3(a)(4)

"A radiocommunication service for the purpose of self-training, intercommunication and technical investigations carried out by amateurs, that is, duly authorized persons interested in radio technique solely with a personal aim and without pecuniary interest."

Summary:

You can transmit signals when you pass the ham radio exams and get the license in certain frequency bands. You can't do any business over ham radio communication. You have no privacy on ham radio communication contents.
Ham radio's role in technology

The ancestor of "online chatting" for everyone

Morse Code: "The original digital" (of 10~20bps!)

Emergency backup communication volunteers

Contributing a lot for disseminating electrical and electronic technologies to the general public

Ham radio ops are the ancestor of nerds/geeks/etc.

http://www.flickr.com/photos/andertoons-cartoons/3032153812/

(myself, circa 2004)
Amateur radio has no privacy

You must identify the callsign at least every 10 minutes and at the end of communication.

Your mailing address is open in the FCC CORES database associated with your callsign.

The contents transmitted over amateur radio is not protected by the law (47 CFR 605 (a)), so they can be used, disclosed, or analyzed for any purposes by any means.

Good for data analysts: ham radio contents (at least in the USA) are a very good source of training data.
Shortwave and ionosphere

CW Skimmer for Morse Code

Morse Code is automatically decodable
CW Skimmer: spotting and reporting "CQ" messages

Courtesy: Alex Shovkoplyas, VE3NEA
http://www.dxatlas.com/cwskimmer/
reversebeacon.net

Beacon: sending periodic signals for measurement
Reverse beacon: listening to signals and sending reports to a hub system for the data collection
Gathering data from CW Skimmers around the world

Each report includes the location of the reporter (de) and the reported station (dx), where (freq) and when the signal is heard, including the strength (snr) and sending speed.

<table>
<thead>
<tr>
<th>de</th>
<th>dx</th>
<th>freq</th>
<th>cq/dx</th>
<th>snr</th>
<th>speed</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>V51YJ</td>
<td>KH6MB</td>
<td>14017.0</td>
<td>CW CQ [LoTW]</td>
<td>15</td>
<td>31</td>
<td>0230z 07 Mar</td>
</tr>
<tr>
<td>K1TTT</td>
<td>TX5K</td>
<td>14033.1</td>
<td>CW CQ</td>
<td>31</td>
<td>30</td>
<td>0230z 07 Mar</td>
</tr>
<tr>
<td>ZL2HAM</td>
<td>AA6KJ</td>
<td>21016.9</td>
<td>CW CQ [LoTW]</td>
<td>21</td>
<td>20</td>
<td>0229z 07 Mar</td>
</tr>
<tr>
<td>KH6LC</td>
<td>AA6KJ</td>
<td>21016.9</td>
<td>CW CQ [LoTW]</td>
<td>27</td>
<td>20</td>
<td>0229z 07 Mar</td>
</tr>
<tr>
<td>K8ND</td>
<td>NS6C</td>
<td>14022.4</td>
<td>CW CQ</td>
<td>7</td>
<td>23</td>
<td>0229z 07 Mar</td>
</tr>
<tr>
<td>NU6O</td>
<td>KK4BMB</td>
<td>14018.5</td>
<td>CW CQ [LoTW]</td>
<td>19</td>
<td>15</td>
<td>0229z 07 Mar</td>
</tr>
</tbody>
</table>

Source: http://www.reversebeacon.net/
WSPR: low-bandwidth protocol

50 bits in TWO MINUTES for callsign, location, power
4-FSK, 1.4648 baud, ~6Hz BW, 162 symbols, ~110.5sec
Designed for low-power ionospheric beacons
"Weak Signal Propagation Reporter"
Designed by Nobel Laureate Prof. Joe Taylor, K1JT

- Runs on Windows/Linux/FreeBSD/OSX
- Protocol is open
- Implementations on stand-alone transmitters widely available (only time synchronization is required)
- No need for a high-profile system; a simple antenna and a 5-watt transmitter works from Japan to Norway (my case)

WSPR protocol and software details at:
http://physics.princeton.edu/pulsar/K1JT/wspr.html
WSPRnet spot database

Realtime WSPRnet spot database:
http://wsprnet.org/drupal/wsprnet/spots

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Call</th>
<th>MHz</th>
<th>SNR</th>
<th>Drift</th>
<th>Grid</th>
<th>Pwr</th>
<th>Reporter</th>
<th>RGrid</th>
<th>km</th>
<th>az</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-03-07 02:24</td>
<td>K8CYV</td>
<td>7.040119</td>
<td>-15</td>
<td>0</td>
<td>EN63ve</td>
<td>5</td>
<td>K9AN</td>
<td>EN50wc</td>
<td>378</td>
<td>206</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>WG2XJM</td>
<td>0.475713</td>
<td>-9</td>
<td>0</td>
<td>EN91wr</td>
<td>5</td>
<td>N1DYL</td>
<td>FN43</td>
<td>768</td>
<td>72</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>G4CAO</td>
<td>3.594181</td>
<td>-12</td>
<td>0</td>
<td>IO91si</td>
<td>5</td>
<td>G7JVN</td>
<td>JO00gv</td>
<td>86</td>
<td>126</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>K7NVH</td>
<td>10.140117</td>
<td>-10</td>
<td>0</td>
<td>CN87us</td>
<td>5</td>
<td>KL1X</td>
<td>BP51ip</td>
<td>2291</td>
<td>322</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>DC2XX</td>
<td>7.040100</td>
<td>+1</td>
<td>-3</td>
<td>JO53dp</td>
<td>5</td>
<td>W1-7897</td>
<td>FN42da</td>
<td>5905</td>
<td>293</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>KE7TYT</td>
<td>7.040098</td>
<td>+3</td>
<td>-1</td>
<td>DN40bv</td>
<td>5</td>
<td>W8QYT/7</td>
<td>DM33</td>
<td>831</td>
<td>188</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>K7NVH</td>
<td>10.140119</td>
<td>-21</td>
<td>0</td>
<td>CN87us</td>
<td>5</td>
<td>K5XL</td>
<td>EM12kp</td>
<td>2698</td>
<td>119</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>K8CYV</td>
<td>7.040118</td>
<td>-4</td>
<td>0</td>
<td>EN63ve</td>
<td>5</td>
<td>W8QYT/7</td>
<td>DM33</td>
<td>2563</td>
<td>254</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>W3BCW</td>
<td>10.140149</td>
<td>-3</td>
<td>0</td>
<td>FM19ka</td>
<td>5</td>
<td>K5XL</td>
<td>EM12kp</td>
<td>1932</td>
<td>255</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>WA3UTC</td>
<td>10.140193</td>
<td>-9</td>
<td>1</td>
<td>FM05or</td>
<td>1</td>
<td>K5XL</td>
<td>EM12kp</td>
<td>1718</td>
<td>264</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>DC2XX</td>
<td>7.040104</td>
<td>+6</td>
<td>-1</td>
<td>JO53dp</td>
<td>5</td>
<td>LZ2RKG</td>
<td>KN12qq</td>
<td>1553</td>
<td>136</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>DL1FX</td>
<td>3.594118</td>
<td>-24</td>
<td>0</td>
<td>JN49gr</td>
<td>0.2</td>
<td>DK9LO</td>
<td>JO54ch</td>
<td>522</td>
<td>12</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>N6SPP</td>
<td>14.097029</td>
<td>-26</td>
<td>-2</td>
<td>BP51cf</td>
<td>2</td>
<td>W7QL</td>
<td>DN40bo</td>
<td>3427</td>
<td>114</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>WG2XJM</td>
<td>0.475722</td>
<td>+2</td>
<td>0</td>
<td>EN91wr</td>
<td>5</td>
<td>KA9CFD</td>
<td>EN40om</td>
<td>903</td>
<td>265</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>K7NVH</td>
<td>10.140117</td>
<td>-19</td>
<td>1</td>
<td>CN87us</td>
<td>5</td>
<td>KC6KGE</td>
<td>DM05gd</td>
<td>1423</td>
<td>169</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>WW6D</td>
<td>144.490520</td>
<td>+13</td>
<td>0</td>
<td>CM88pl</td>
<td>20</td>
<td>N6GN</td>
<td>CM88ok</td>
<td>9</td>
<td>237</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>W3BCW</td>
<td>10.140146</td>
<td>-8</td>
<td>1</td>
<td>FM19ka</td>
<td>5</td>
<td>KC6KGE</td>
<td>DM05gd</td>
<td>3746</td>
<td>277</td>
</tr>
<tr>
<td>2013-03-07 02:24</td>
<td>WA3UTC</td>
<td>10.140189</td>
<td>-11</td>
<td>1</td>
<td>FM05or</td>
<td>1</td>
<td>KC6KGE</td>
<td>DM05gd</td>
<td>3658</td>
<td>281</td>
</tr>
</tbody>
</table>
Realtime WSPRnet map is available at: http://wsprnet.org/drupal/wsprnet/map
How big is the data?

Reversebeacon.net
   ASCII CSV per day, 11 fields/record, ~70 bytes/record
   21-FEB-2009 - 9-MAR-2013: ~171M records

WSPRnet
   ASCII CSV per month, 15 fields/record, ~100 bytes/record
   11-MAR-2008 - 9-MAR-2013: ~125M records

Total size of the above data
   ~4GB compressed
   ~21GB uncompressed

Can RDBMS scale well for this size? - Maybe not.
So I decided to use...

Disclosure: I am employed by Basho Japan KK, a group company of Basho Technologies, Inc., the creator and developer of Riak, an open-source distributed database that provides extreme high-availability, fault-tolerance, and operational simplicity, written in Erlang/OTP.
Why Riak?

It's written in Erlang/OTP

  We're in the Erlang Factory, aren't we?

It's what I've been working for since February 2013

  Disclaimer: I'm still learning a lot of things about it right now; contents of this presentation is far from complete

It's open source - https://github.com/basho/riak/

It scales - the more data you need to handle, the more nodes added will nicely crunch the data

  This means prototyping is easy; try first by a smaller subset of the dataset, then deploy it to the larger datasets
Database structure for Riak

**a)** Only storing per-month/day CSV
- Pros: fast, lighter load on Riak servers
- Cons: The clients have to search individual CSV files

**b)** Storing each event record with indexing
- Pros: searchable on Riak, as if using an SQL DBMS
- Cons: heavy load on Riak servers for preprocessing

I decided to choose **b)** because:
- Riak has the search ready (Riak Search, Yokozuna)
- Riak can handle JSON, easily convertible from CSV
- Little time was available for prototyping external programs
- I wanted to explore what I can do with Riak alone
My testing platform(s)

My home Acer laptop (AS3830T-N54D)
   The same one I used for TinyMT at Erlang Workshop 2012
Intel Core i5-2410M quad-core CPU, 2.3GHz clock,
   8Gbyte RAM, 500Gbyte 2.5" HDD (a notebook PC)
FreeBSD/amd64 9.1-STABLE r247012
   With only 8Gbytes of the swap file space
Erlang/OTP R15B03-1
   Unfortunately as of March 2013, Riak does not run on R16B
Note: Riak is supported only on R15B01 within R15s yet
Supplemental test environment:
   MacBook Air 11" with OSX 10.8.2
Compiled R15B03-1 using kerl
   kerl: allows multiple versions of Erlang/OTP instances, easily switchable for each shell environment, compatible at least for FreeBSD, Ubuntu, and OSX
   https://github.com/spawngrid/kerl
Installed vanilla Riak 1.3.0
First, run 5 nodes, with default parameters (w=3)
   Bitcask storage backend
   This resulted in swap space exhaustion
Running 3 nodes seemed to be the feasible maximum
   Since w=3, this will give complete copies for each node
What I did first (2/2)

Tested Riak Search

Solr-like interface for Riak
Conversion from JSON fields to searchable indexes
Suitable for CSV, after converting CSV to JSON
https://github.com/basho/riak-python-client
Must build index for each bucket before putting in the data
search-cmd install [bucket-name]

Riak Search worked, but...

It does not implement full range query and hard to use
It is slow and consumes a lot of CPU time on BEAM
Getting the numbers of each field member is unsupported
The xload here -> shows a weird periodic change of the load average on the testing system; the period is ~15 to 20 minutes. The system is virtually idle; no Riak server access from the client programs.
So I decided to try Yokozuna

A fast full-text search for Riak with Apache Lucene

Yokozuna stands for the highest ranking of Sumo tournament; once you become a Yokozuna, the only way to get demoted is to resign, so the word Yokozuna is only given to the best and the brightest performers.

"Everybody can concentrate at least ten minutes in a day; just for ten minutes!"

-- Futabayama Sadaji
(translated by Kenji Rikitake)

<- Unryu-style Tsuna (rope)

The 35th Yokozuna
Futabayama Sadaji

Source: Wikimedia Commons
http://en.wikipedia.org/wiki/Futabayama_Sadaji
Running Yokozuna (1/3)

Yozokuna has its own Github repository

https://github.com/basho/yokozuna

A Riak branch automatically install Yokozuna

Branch yz-merge-1.3.0 (As of 17-MAR-2013)

Requires Apache Ant later than version >= 1.8.2

Running Apache Solr 4.1.0 (As of 17-MAR-2013)

Index must be created for each bucket

  curl -XPUT -i -H 'content-type: application/json'
  http://localhost:10018/yz/index/name_of_index
Running Yokozuka (2/3)

For crunching JSON records:

Set Content-Type to application/json

To tell data type for each JSON field, an easy way is to use type-specific suffixes for the field names, e.g., spotid_s for a string (_s), supported by the default schema

https://github.com/basho/yokozuna/blob/master/priv/default_schema.xml

Java language system is required

For FreeBSD, install Port java/openjdk6 or openjdk7

Parallelizing PUTs is essential for better performance

GNU parallel will be the easiest way for a shell script

http://www.gnu.org/software/parallel/
Running Yokozuna (3/3)

Using NIF for Bitcask enabled
   Stability unchanged
   didn't affect much on the writing performance
   (probably indexing needs most of the CPU time?)

Protocol buffer used for the Python clients
   More lightweight and faster than HTTP, especially for
   smaller objects for each PUT operation

Choice of keys
   Reversebeacon: let Riak assign the keys (=random)
   WSPRnet: used unique Key in the CSV
   No performance/usage change between the two
Yokozuna PUT test results

3-day data of 1-3 January 2013
  Reversebeacon.net: 547371 records
  WSPRnet: 456903 records
  riak-admin backup result: ~2.24Gbytes in 170 seconds for all 3 nodes in the cluster (13.18Mbytes/sec)

PUT speed: ~270 records/sec with 2 parallel clients

Other notes
  Loaded Riak serves may cause timeouts

  make devrel copies ALL of .jar files for each node; make stagedevrel uses much less disk space
WSPR query example

curl "http://example.com:10018/search/records_yz_wspr? # bucket name
wt=json& # Returning JSON
indent=on& # JSON result: indented
rows=0& # No raw result rows returned
q=band_i:21& # Pick up all records with band_i=21
facet=true& # Faceted search
facet.field=freq_d& # Faceted field: freq_d
facet.sort=index& # Sort by freq_d value
facet.mincount=1" # omit zero entry values
WSPR query result example

```json
{
    "responseHeader": {
        "status": 0,
        "QTime": 9,
        "params": { (skipped), "rows": "0" },
        "response": {
            "numFound": 3890,
            "start": 0,
            "maxScore": 5.7686505,
            "docs": []
        },
        "facet_counts": {
            "facet_queries": {},
            "facet_fields": {
                "freq_d": { (skipped),
                "21.096001": 4,
                "21.096002": 18,
                "21.096003": 16,
                "21.096004": 9,
                "21.096005": 9,
                "21.096006": 20,
                "21.096007": 11,
                "21.096008": 36,
                (skipped),
                "21.096083": 33
            },
            "facet_dates": {},
            "facet_ranges": {}
        }
    }
}
```
Plotting the query result
Reverse Beacon example

Each record contains

Which continent the spots are from (de_cont_s)
Which continent the spotted stations are (dx_cont_s)

A simple command to search per each de_s:

curl -s ' http://127.0.0.1:10018/search/records_yz_rb?wt=json&indent=yes&rows=0&facet=true&facet.field=dx_cont_s&facet.sort=index&q=de_cont_s:SA'
## RB continental spots

(1-3 January 2013)

<table>
<thead>
<tr>
<th></th>
<th>DX Africa</th>
<th>DX Asia</th>
<th>DX Europe</th>
<th>DX North America</th>
<th>DX Oceania</th>
<th>DX South America</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Africa</td>
<td>147</td>
<td>185</td>
<td>791</td>
<td>651</td>
<td>34</td>
<td>307</td>
</tr>
<tr>
<td>From Asia</td>
<td>16</td>
<td>15550</td>
<td>1709</td>
<td>489</td>
<td>984</td>
<td>53</td>
</tr>
<tr>
<td>From Europe</td>
<td>3889</td>
<td>15573</td>
<td>281141</td>
<td>16325</td>
<td>763</td>
<td>1071</td>
</tr>
<tr>
<td>From North America</td>
<td>1960</td>
<td>3773</td>
<td>47334</td>
<td>137557</td>
<td>1925</td>
<td>4207</td>
</tr>
<tr>
<td>From Oceania</td>
<td>11</td>
<td>1252</td>
<td>820</td>
<td>1608</td>
<td>618</td>
<td>81</td>
</tr>
<tr>
<td>From South America</td>
<td>212</td>
<td>310</td>
<td>1685</td>
<td>3043</td>
<td>83</td>
<td>1141</td>
</tr>
</tbody>
</table>
Lessons learned

Start from smaller datasets

Riak and Yokozuna scales well on large data
For faster prototyping, smaller datasets require less memory and less loading time
After complete writing the processing scripts, they are equally applicable to the larger datasets
Laptops are NOT as powerful as dedicated servers!

Use the most suitable tool for each task

Erlang/OTP has a strength on scalability
Python has a strength on the rich library and fast coding
R is a convenient system for rapid graph drawing
Acknowledgments

Reversebeacon.net devops
WSPRnet devops
Ryan Zezeski of Basho Technologies
  The principal developer of Yokozuna
Basho engineers
  John Caprice, Russell Brown, Jared Morrow
Basho Japan engineers
  Kota Uenishi, Kazuhiro Suzuki, Shun'ichi Shinohara
Basho Japan managers
  Sam Takagi, Takuya Kamakura
BashoWest for their office facility
Thanks

Questions?

More examples:
https://gist.github.com/jj1bdx/5180721