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:

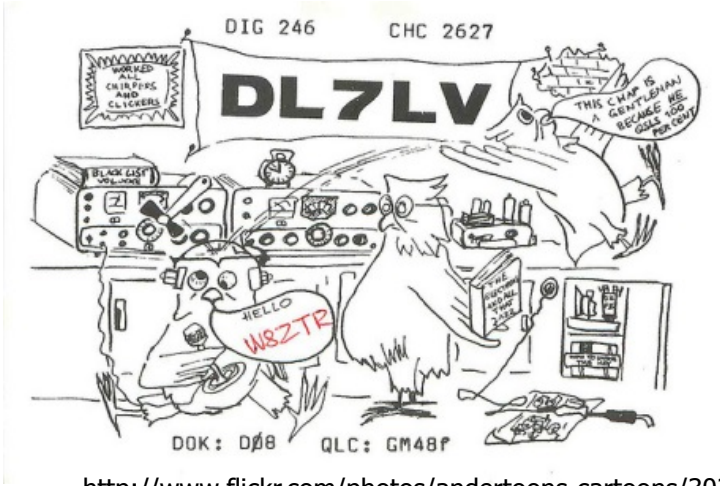
You can transmit signals when you pass the ham radio exams and get the license in certain frequency bands.

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

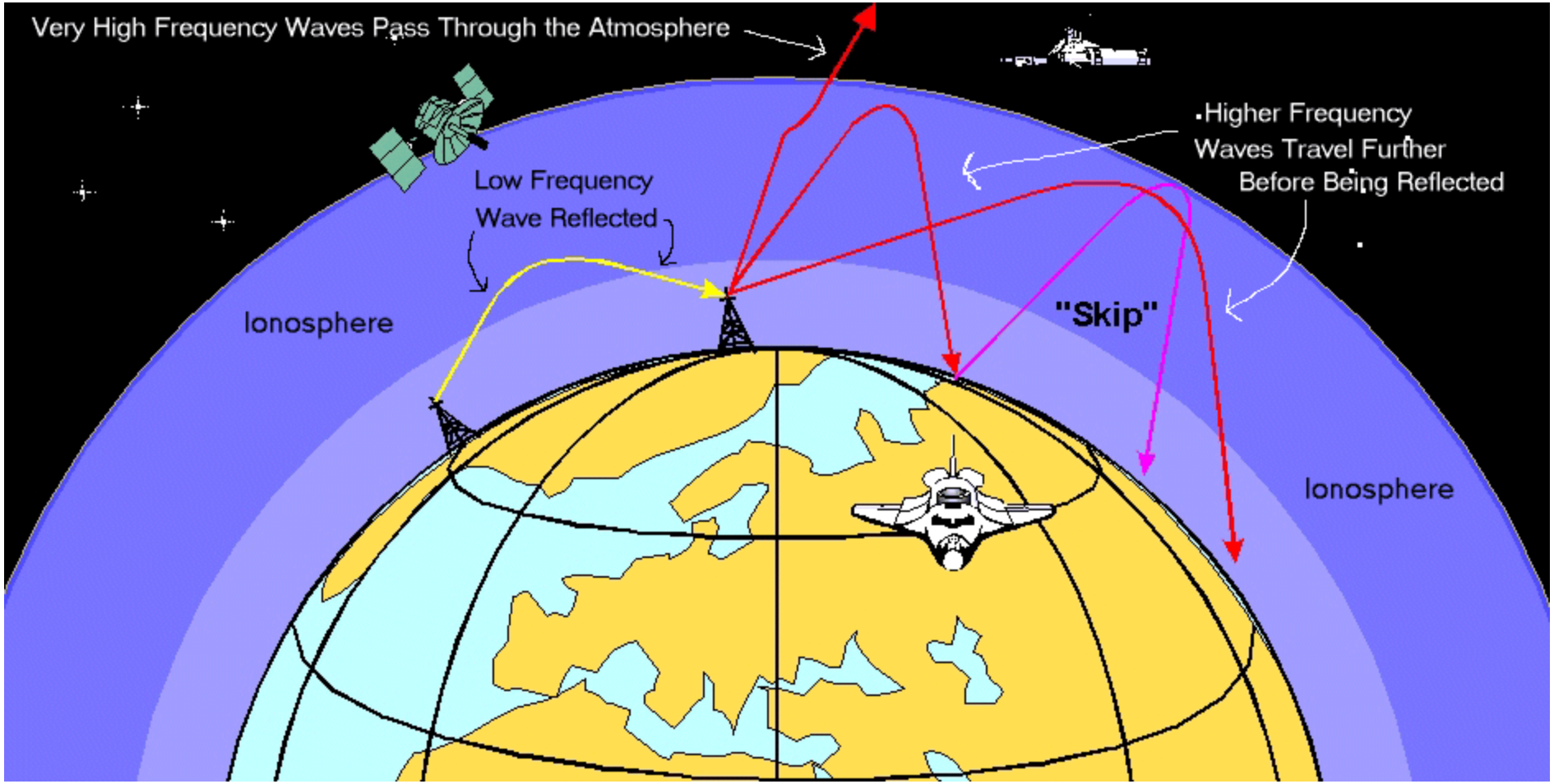
You must identify the callsign at least every 10 minutes and at the end of communication

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

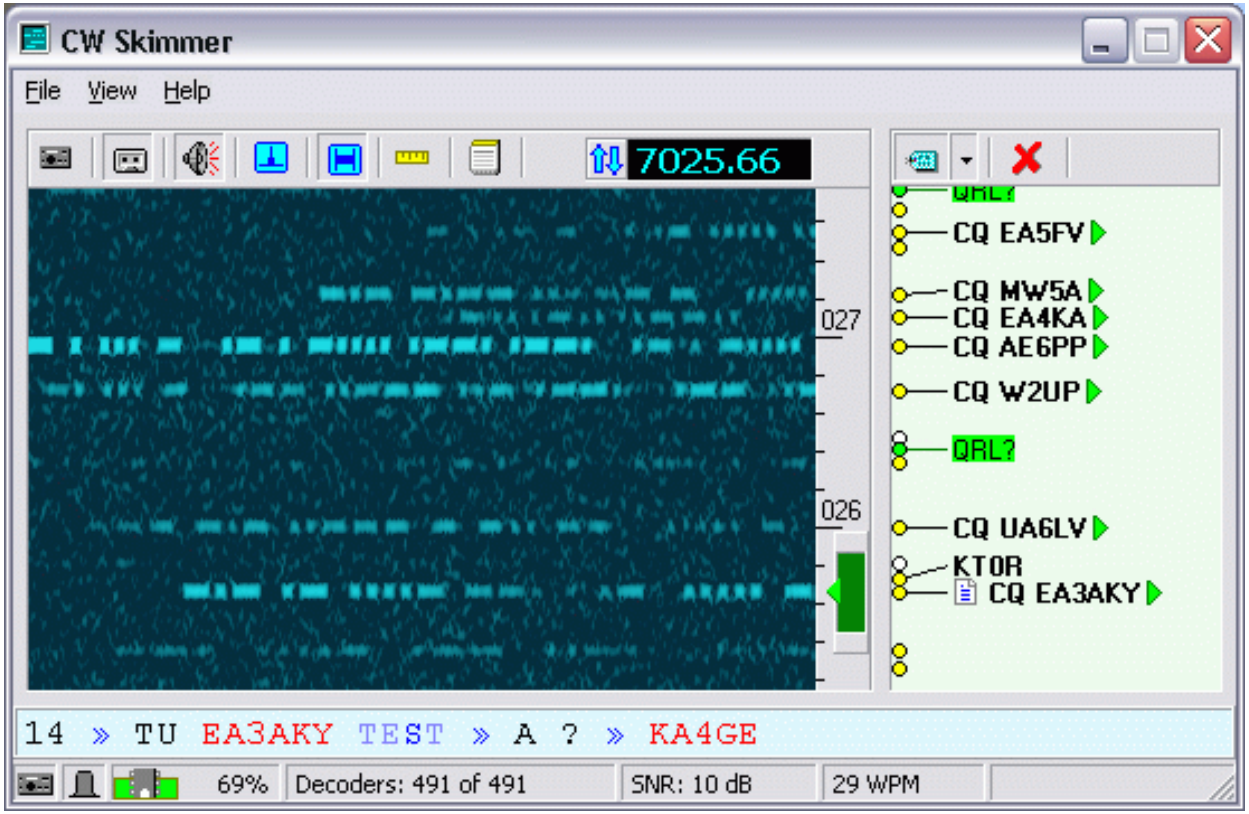


Source: NASA http://radiojove.gsfc.nasa.gov/class/educ/radio/tran-rec/exerc/iono.htm

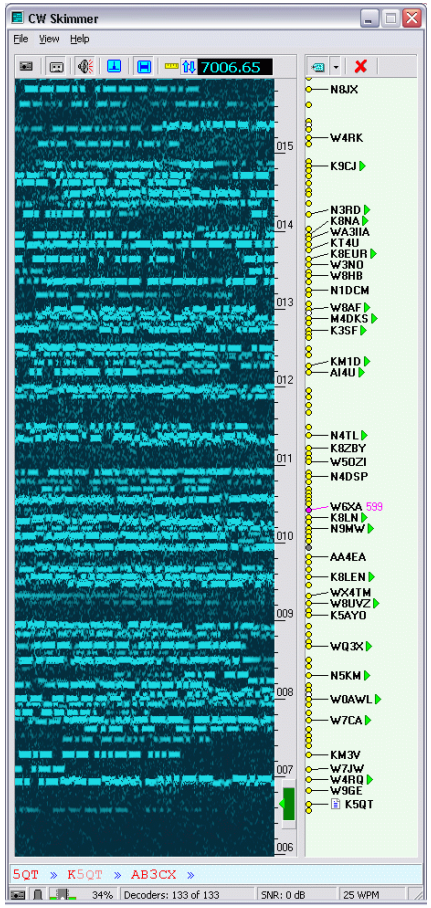
CW Skimmer for Morse Code

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.

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

Source: http://www.reversebeacon.net/

WSPR: low-bandwidth protocol

50 bits in TWO MINUTES for callsign, location, power

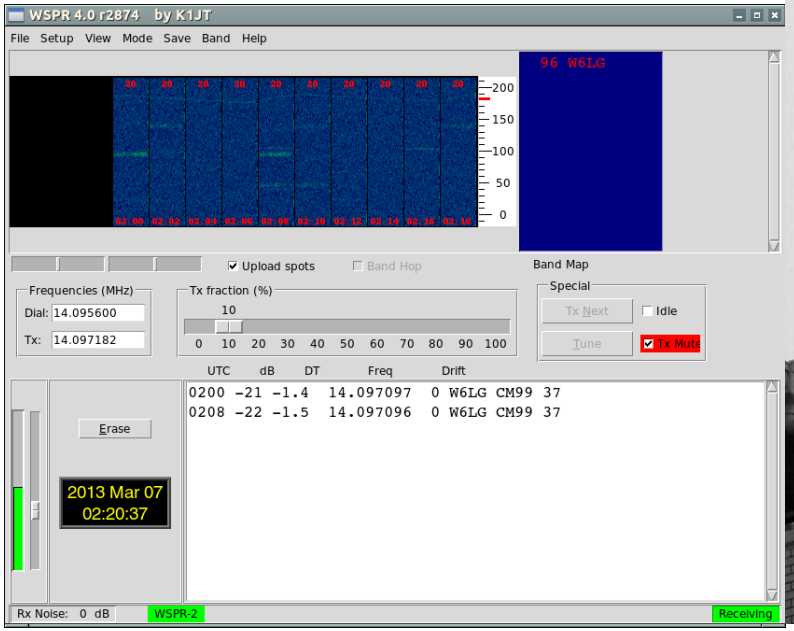
4-FSK, 1.4648 baud, ~6Hz BW, 162 symbols, ~110.5sec

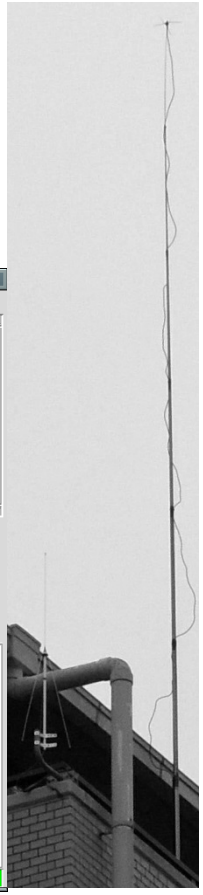
Designed for low-power ionospheric beacons

"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

Spot Database

Specify query parameters

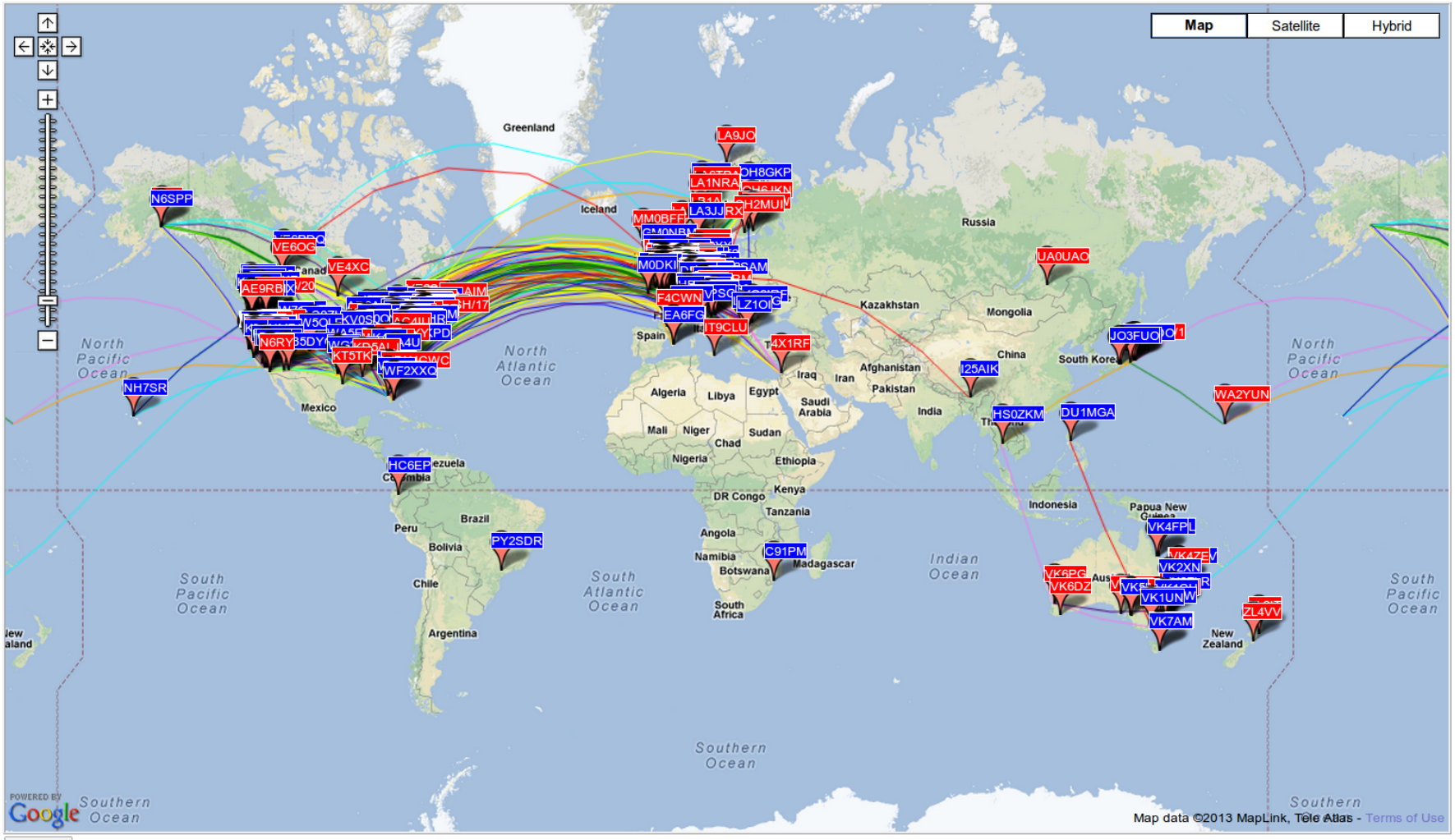
50 spots:

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

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

WSPRnet propagation map

Propagation Map



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

What I did first (1/2)

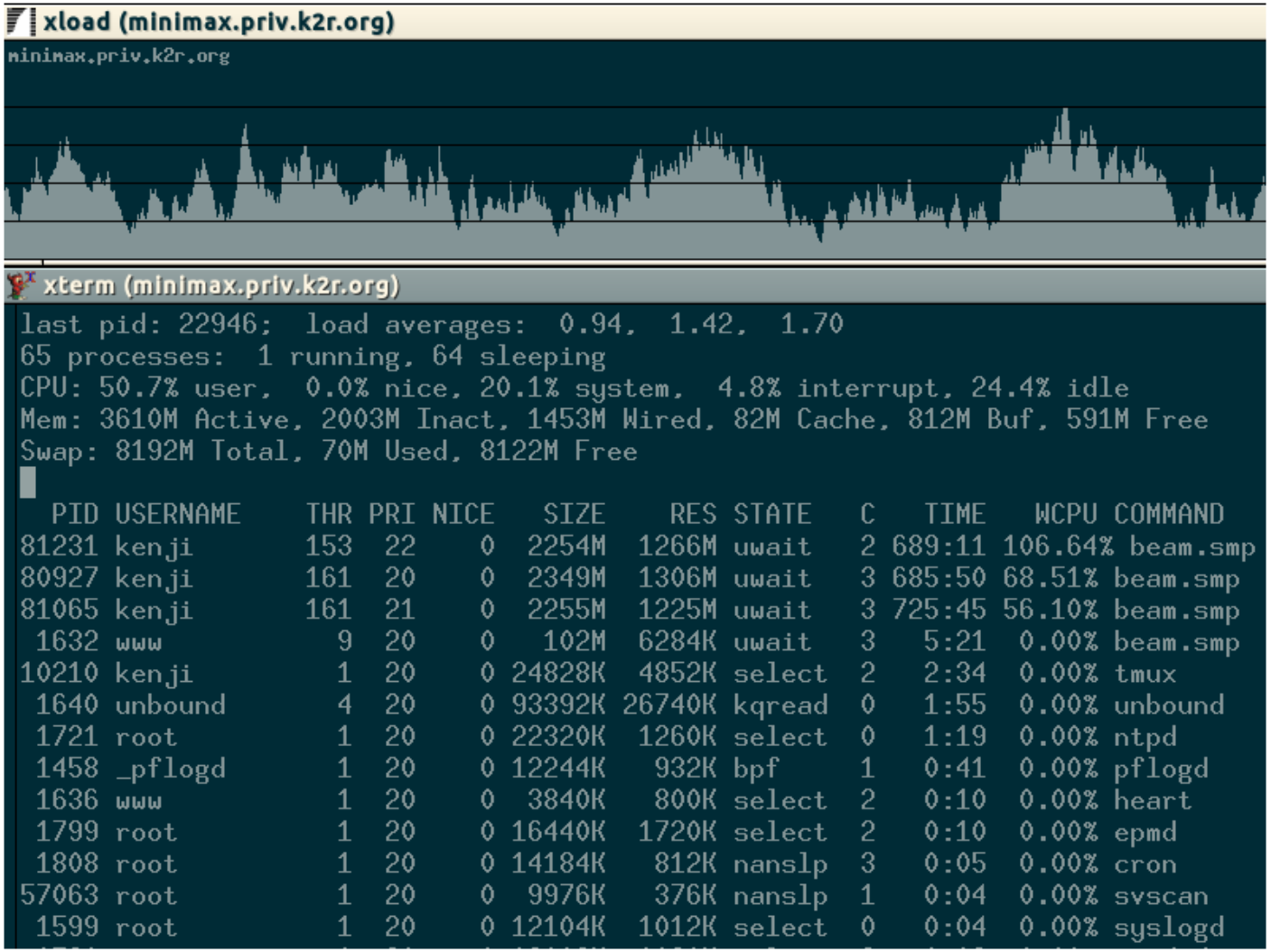
- Compiled R15B03-1 using kerl
  - kerl: allows multiple versions of Erlang/OTP instances, easily switchable for each shell environment, compatible at least for FreeBSD, Ubuntu, and OSX
  - https://github.com/spawngrid/kerl
- Installed vanilla Riak 1.3.0
- First, run 5 nodes, with default parameters (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

Riak Search's weirdness

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

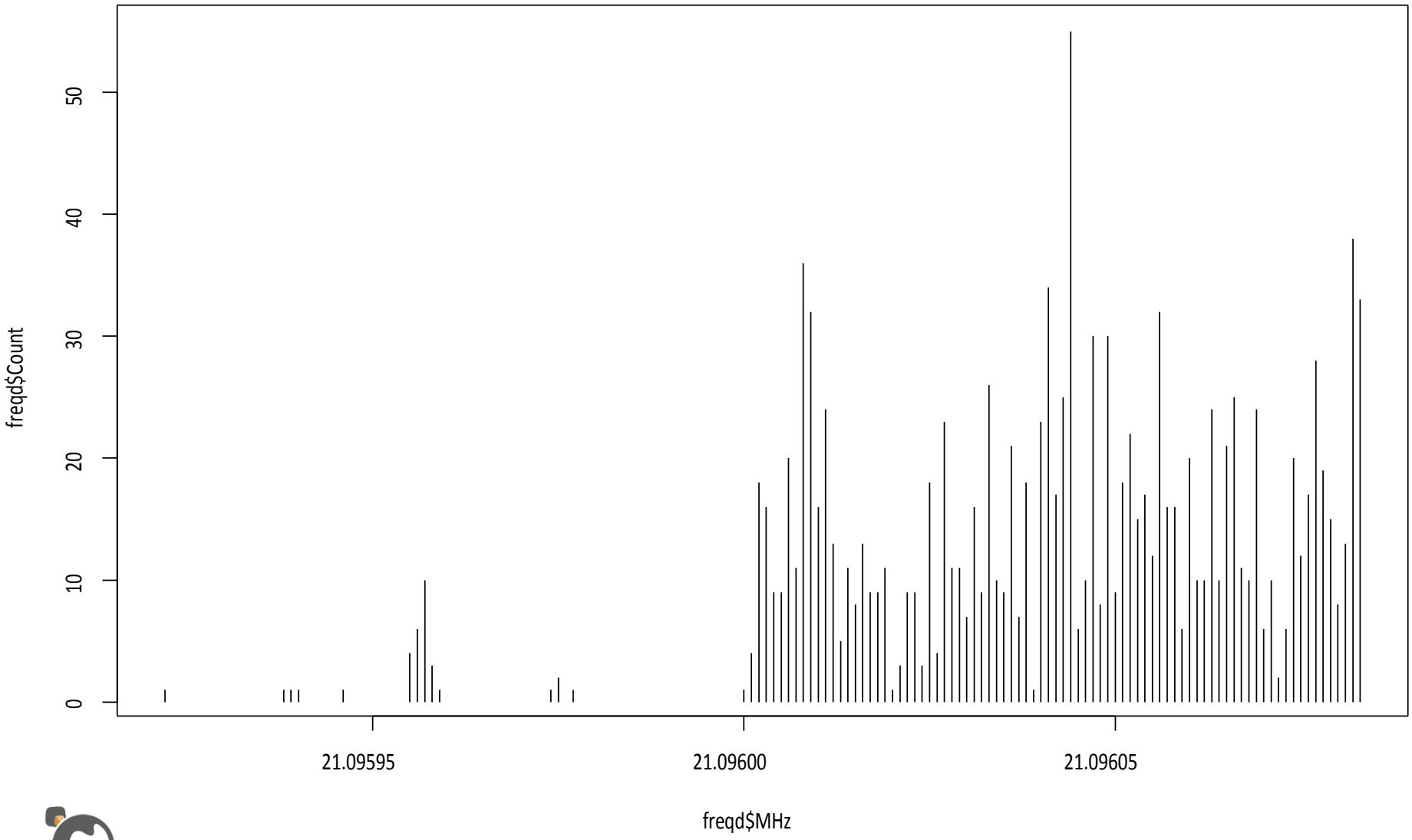
WSPR query example

```
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


```
{ "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)

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

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