Aids to the Presentation and Analysis of WSPR Spots: TimescaleDB database and Grafana

Gwyn Griffiths, G3ZIL and Rob Robinett, AI6VN

Abstract – The wsprnet.org database provides a very good service for collecting, and making available, some 2.4 million spots from about 2500 reporters on a typical day. Its web page query tool, and those of third parties that scrape data from wsprnet.org, fulfill the needs of very many users. However, for users seeking to glean additional information from their own WSPR spots, or from those of a wider community, the tools provided by a relational database and a data visualization package become necessary. This paper outlines the rationale behind the WsprDaemon time series database, our initial experience with Influx as the database, and the reasons for moving to TimescaleDB. The system's architecture is described, highlighting resilient data gathering with user and server caches, an ability to handle delayed spot reporting, and a close coupling to Grafana as the visualization package. Three examples of Grafana Dashboards illustrate this approach and the utility of the results in providing users with a richer set of graphics to help them understand what WSPR spots tell them about propagation and their own installations and noise environment.

1. INTRODUCTION

Since the introduction of the Weak Signal Propagation Reporter (WSPR) protocol in 2008 reception reports have been uploaded to **wsprnet.org**. Originally written by Bruce Walker, W1BW, and now maintained by a small team of volunteers, wsprnet.org provides a simple, web-based interface of pull-down options to query its SQL database. There is also the 'old database' interface at **wsprnet.org/olddb** that supports web-page queries and 'scraping' using curl, a command line tool for data transfer to or from web pages (**curl.haxx.se**). In addition, there is an invitation-only Application Programming Interface (API) by the wsprnet administrator Gary McMeekin, W1GJM that returns JavaScript Object Notation (JSON) data [1].

Writing in August 2010 Taylor and Walker reported that the WSPR database comprised 32 million spots, with 300-500 stations reporting 50,000 to 100,000 spots daily [2]. It is a tribute to the early work, the on-going support and investments in hardware that wsprnet.org provides an effective central reporting database for some 2500 reporting stations handling 2.4 million spots a day. Harder to quantify is the load on the database from its web-page users and from several third-party scraper and API applications.

Third-party applications come in several forms. There are near-real time [3], daily [4] and monthly [5] ranked lists, a map with simple pull-down options [6], suites of tables, charts and maps as a website [7], an app for mobile devices [8], to a comprehensive graphing tool drawing on its own copy of the full wsprnet.org database [9].

Given wsprnet.org itself and the plethora of third-party applications, why did we see the need for a separate database and graphing tool? There were several drivers, in part cascading as the initial project proved useful:

1. Author Robinett's concept of a robust and reliable reporting tool for WSPR spots from the multi-channel KiwiSDR (**www.kiwisdr.com**) led to him writing a Linux application, WsprDaemon (**wsprdaemon.org** and [10]), now with over 40 users.
2. Recognizing that the KiwiSDR was capable of estimating noise level at its input author Griffiths contributed an investigation of noise estimation at the same time and on the same frequencies as receiving WSPR spots [11]. As that data could not be reported to wsprnet.org a separate database was needed. Tommy Nourse, KI6NKO, suggested using Influx, a database specifically designed for time series data, with Grafana as the graphical display.
3. Following encouragement from several members of the HamSci community (**hamsci.org**) we added spots from WsprDaemon users to an Influx

Gwyn Griffiths is an independent amateur and citizen scientist based in Southampton, U.K. (corresponding author e-mail: gwyn@autonomousanalytics.com).

Rob Robinett based in Berkeley California, is CEO of TV equipment manufacturer Mystic Video and he's founded a series of Silicon Valley startups. He recently "rediscovered" amateur radio - after an absence of more than 40 years - and he applies his software expertise to developing systems to measure short wave radio transmission conditions.

(**www.influxdata.com**) database, and as its usefulness became apparent we added spots for all reporters, obtained from wsprnet.org.

4. Over time, an ambition grew to be able to offload some of the third party data scraping from wsprnet.org by providing a secondary source of reported spots.

The remainder of this paper is organized as follows: section 2 introduces time series databases, summarizes our experience of Influx, and the reasons for our move to TimescaleDB (**www.timescale.com**); section 3 describes our TimescaleDB implementation, its capability and current user interfaces; section 4 describes how Grafana links to TimebaseDB and the features available; section 5 provides examples of insights into propagation and station performance that can be obtained from different graphical representations of the basic and derived data, section 6 shares some thoughts for future development of WsprDaemon, the database and graphical visualization.

2. TIME SERIES DATABASES: INFLUX AND TIMESCALEDB

Time series databases have emerged as a class of tools that deliver performance improvements over traditional databases by recognizing several characteristics of many time series. These include many insert operations, often in batches, and often timely, that is, few inserts are delayed, updates to existing inserts are rare, and queries often specify a time interval [11]. Time series databases 'chunk' the data in time, keeping the most recent data in memory. This facilitates fast response times for simple queries on recent data but requires paging between memory and disk for data in older chunks, slowing the response. In addition to Influx and TimescaleDB others include Clickhouse, OpenTSDB, Riak TS, and Gorilla. As database novices we were steered toward Influx.

2.1 Influx time series database

Influx is a purpose-built time series database with its own query language, similar to SQL. Using its excellent documentation [12] our database was set up in hours. The downside was that it took us time and accumulated experience to become aware of the limitations of Influx's approach for the characteristics of the WSPR dataset.

Briefly, Influx creates a 'measurement' (similar to an SQL table) with indexed tags (character fields such as Callsigns and Locators, but also Band, as only tags can be queried) and non-indexed fields (numeric fields such as SNR or Drift). Data are not stored as simple rows as they would appear in a spreadsheet or in many databases but are associated with the tag sets.

Cardinality is a term for the number of unique combinations of measurements, tags and fields in a database. As cardinality grows, Influx's response time to queries increases, as does its CPU and memory requirements. It helps that most callsigns are always associated with the same locator. But even so, after four days, with 5.8 million spots, the cardinality had reached over 256,000. Influx's documentation considered this 'moderate' suggesting hardware with 4-6 cores and 8–32GB memory. As we were running on a Digital Ocean Droplet with 2 cores and 4GB memory the performance had become unacceptable; the time taken to return the results of a single-user simple query to list 10,000 spots increased from a barely acceptable 15s to 28s.

While we were prepared to move to higher-performance hardware, the continued, albeit reducing, rise in cardinality, with no sign of reaching a stable value, Figure 1, led us to look for an alternative database. Potential users of Influx should carefully consider the likely cardinality and associated hardware requirements for their own particular requirements.

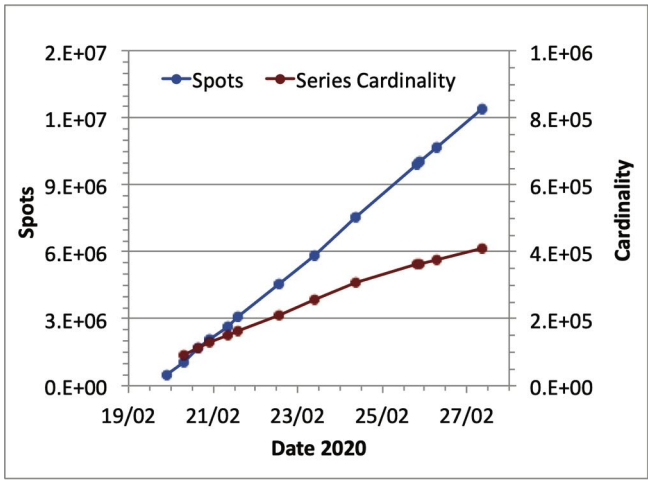

Figure 1. An almost linear rise in the number of spots in our Influx database gave rise to a cardinality that was still rising after 7 days. The cardinality : spots slope of 0.036 after 1 day had reduced to 0.018 after 7 days, but its continued rise would give unacceptable query times on the available hardware.

2.2 Why a TimescaleDB time series database

Further reading, our experience with Influx giving us a better insight into what was needed, and a readable, fair-minded comparison of Influx and TimescaleDB [12] led us to TimescaleDB. TimescaleDB is an extension to the very well established open-source relational database postgreSQL (**www.postgresql.org**) that optimizes its performance with time series. It B-tree approach to

indexing data promised a better ability to handle the high cardinality inherent in WSPR data.

Importantly, as a relational database, TimescaleDB enables join operations both within individual tables (self-join) and between tables. This feature was not available in v1.7 Influx that we were using. Join operations allow queries such as: calculate the difference in SNR for the same sender at the same time in the same band for two different reporters.

3. Implementing a TimescaleDB database

3.1 Data architecture overview

The data architecture of our August 2020 implementation is shown in Figure 2. In brief:

1. WsprDaemon client software on the reporting station's computer caches spot data, to avoid gaps during outages. Records comprising the normal WSPR fields are forwarded using HTTP Put to wsprnet.org and an extended set by FTP to logs.wsprdaemon.org, including estimates of local noise if enabled.
2. WsprDaemon server logs.wsprdaemon.org has two spots input paths: direct from WsprDaemon users, and from all users from wsprnet.org via its API every two minutes. Preprocessing adds fields including numeric latitude, longitude and azimuth at the receiver as useful variables to plot when visualizing data [Section 5].
3. Spots from WsprDaemon users with additional fields are inserted into table wsprdaemon_spots and their noise data goes into table wsprdaemon_noise in the TimescaleDB database. Spots arriving via the wsprnet.org API go to table spots. Users connect to this database to query data. At current loads the server is well able to handle data insertion and user queries.
4. Data is also obtained from third parties via a scraper or an API. Currently we obtain the three-hourly geomagnetic disturbance index Kp from the US Space Weather Prediction Center.
5. As this data arrives at the server it is cached before being mirrored to a backup server logs1.wsprdaemon.org and inserted into a replica TimebaseDB database.

3.2 Hardware outline

High availability, sufficient memory to hold at least one

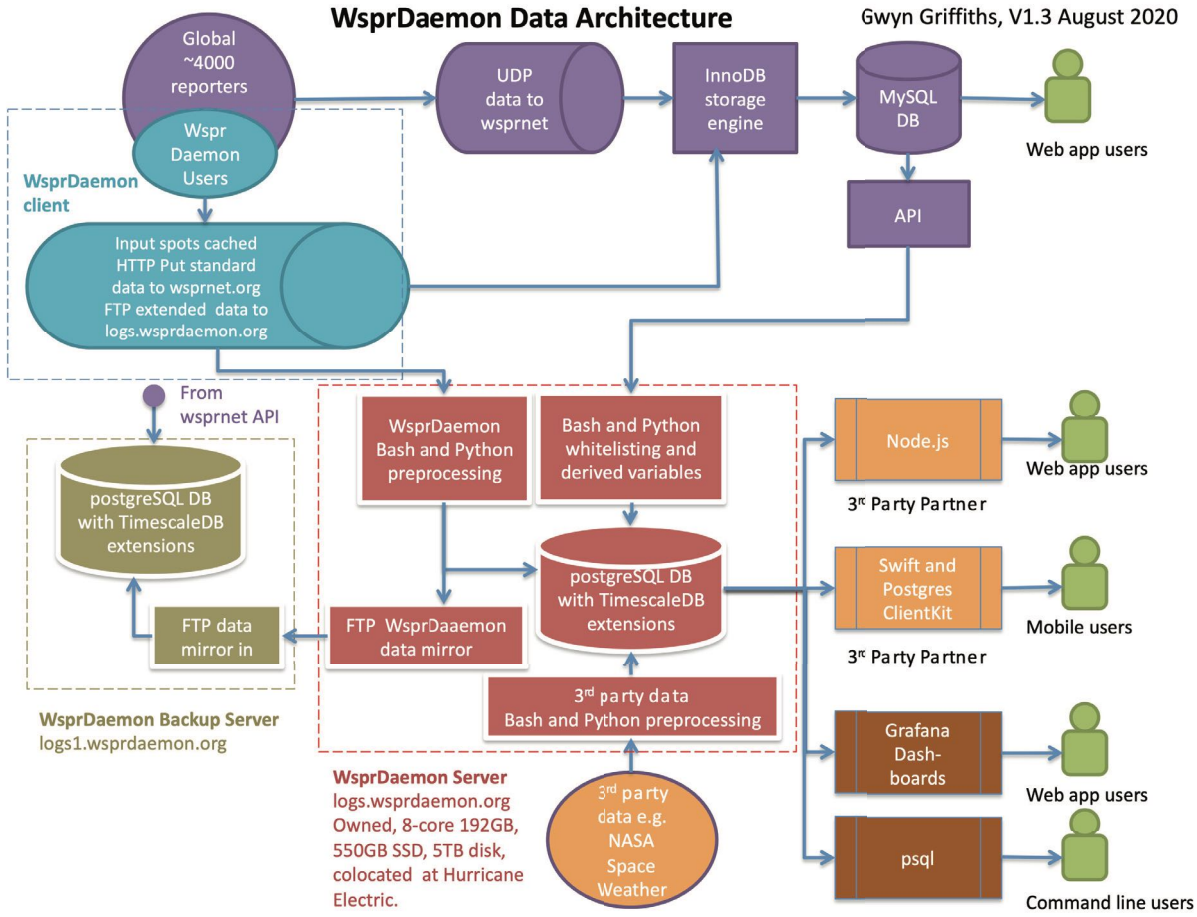

Figure 2. Diagram showing the data architecture of WsprDaemon, its data input pathways, its resilient design in having a server with a TimescaleDB database mirror data to a separate backup.

month of data in memory, appropriate processing power and good Internet connection are key requirements for the server logs.wsprdaemon.org. Currently these requirements are met with an owned Dell 8-core 192GB memory, 550GB SSD and 5TB disk machine. To ensure resilience against power outages and to provide a Gb Internet connection the server is collocated at a Hurricane Electric data center. A backup machine provides a hot-standby facility.

3.3 TimescaledDB installation and database set-up

Open Source, Community and paid-for Enterprise versions of TimebaseDB are available for a range of operating systems, including Windows, MacOS and several Linux distributions [14]. Installation was straightforward, edits to the configuration file were necessary to allow, among others, remote connections, access by password, and to set time zone to UTC. Port 5432 needs to be open on the router/firewall(s).

Initial set-up of a database is well described in the TimescaleDB documentation, although new SQL users will find the tutorial and examples pages at **www.postgresqltutorial.com** very helpful. The TimescaleDB extension need only be loaded once. An interactive program, *psql*, enables effective communication with the database for administrative tasks and to download the results of queries as csv files.

For our initial database we migrated WSPR data from Influx using a purpose-coded tool, Outflux [15]. If one is migrating to TimescaleDB from another postgreSQL database use the psql command pg_dump.

On creation, database tables will simply be postgreSQL tables. The *create_hypertable* TimescaleDB extension converts a postgreSQL table to a hypertable - these are interlinked 'chunk' tables, where a chunk covers a user-set time period. The choice of duration for a chunk needs to consider the incoming average data rate (say in MB/day), the size of the associated index, and the free memory available across all of the active hypertables and databases on the machine. It is advisable that all active chunks fit into 25% of memory.

Early experience showed the spots table from wsprnet.org, with derived variables, grew between 370 and 460MB/day from data, and 58 to 72MB/day from the index. For logs.wsprdaemon.org with 192GB memory we have set a conservative chunk size of 30 days rather than the 83 days implied by the 25% guide. This is to allow us to ensure that there is adequate CPU performance for queries spanning 30 days as our user base grows.

3.4 Query response times

There may be little value in providing examples of query response times without a full, detailed analysis of the load on the server. Nevertheless, given the minimal information in Figure 3, queries via third party app **wsprd.vk7jj.com** on all bands from OE9GHV for up to 50,000 spots over 24 hours returned 33,908 spot lines in 2.3 seconds, and for up to 500,000 spots over 7 days returned 275,784 spot lines in 13.6 seconds. The average time for 33 queries for all spots over the last hour for a variety of stations was 1.4 seconds. [*Note: when preparing this figure logs.wsprdaemon.org was labeled WD1*].

Figure 3. Screen shot of a pg_admin window showing statistics for the logs.wsprdaemon.org server. There are 1 to 8 active sessions; the Tuples in graph shows the Inserts from wsprnet.org; the Tuples out graph shows traffic from user queries. The two peaks at left were for queries via third party app wsprd.vk7jj.com for up to 50,000 spots over 24 hours received by OE9GHV on all bands, the queries returned 33,908 spot lines in 2.3 seconds, the right hand peak, with over 15 million Tuples out was for a query for all spots (275,784) from OE9GHV for 7 days.

4. GRAFANA VISUALISATION WITH TIMESCALEDB

Grafana (**grafana.com**) is a multi-platform open source data visualization package with extensive support for connections to numerous databases including postgreSQL and TimescaleDB. In its terminology the user creates a 'Dashboard' comprising one or more 'panels', a panel may be a time series graph, a simple gauge, a digital readout, a map, or a host of other data representations. In this paper we focus on using Grafana time series graphs with WSPR data from our TimescaleDB database.

We have set up a few read-only example Dashboards at **logs.wsprdaemon.org:3000** with access information available at **wsprdaemon.org/grafana.html.** However, users may want to install Grafana on their own machine to create custom Dashboards. Using Share –> Export options the json code for our Dashboards can be saved and imported into a user's local Grafana instance as starting templates for their own versions.

4.1 Setting up Dashboards and panels

In summary, the steps to creating a data visualization site after installing Grafana are:

- Add a data source: there are currently pull-down options for over 20 sources, including database types, cloud repositories, and enterprise plug-ins. A single Dashboard may draw on data from one or more of these sources. Adding a data source requires connection details: host IP address, userid, password, SSL mode, any connection time limits etc.
- Build a Dashboard: Starting with a new blank panel we add one or more queries if, as with TimescaleDB, the source is a database. After selecting the data source from its known list Grafana provides a comprehensive skeleton SQL Query Builder that the user modifies by deleting unwanted parts, using pull-down options to add specifics, e.g. to select SNR or distance as variables to plot, whether to use aggregate functions, e.g. to count the number of spots received in a time interval, or to form an average. For simple queries the Query Builder is sufficient; more complex queries can be written directly in SQL.
- Choose and customize visualization: Simple graphs of time series are excellent for many variables, but other Grafana options are very useful. Heat maps show the value of a derived quantity, e.g. a count of instances, as a color (effectively a z-axis) within a time period (a time bucket) and over a range of the y-axis variable (bins). Examples of heat maps include the number of spots in 20-minute time buckets and 1000km range bins, or in 15˚ bins of azimuth at the receiver.

5. INSIGHTS INTO PROPAGATION AND STATION

Figure 4. Grafana Dashboard that enables a user to compare distance statistics in 10 minute intervals (lower quartile, median, upper quartile) for two receivers and two bands. As in this example, a comparison may be between two different receivers on the same band, but can also be used for the same receiver on two different bands. Pull down options at top right set the time span and the user can save or export the plot, or save the data in csv or json formats.

In this section we show three graphical examples to illustrate the value of bringing together a time series database and Grafana visualization. A common feature of most of these Grafana Dashboards is that by using template variables the user can easily select, using pull-down options, the stations, bands etc. to plot.

5.1 Insights into propagation - simple time series plots

Figure 4 shows a Dashboard that enables us to look at simple statistics with time of the distances between receiver and transmitters for two receivers and for two bands. The statistics here are the lower and upper quartiles and the median. The two receivers may be the same, but the bands may differ, or, as in this example, we can compare two different receiver stations on the same band.

In this example on 40m, AI6VN/KH6 is on Maui, Hawaii, while KK6PR is in central Oregon. For AI6VN/KH6 the lower quartile represents the distance to the west coast of N. America. Propagation over that path is almost continuous, except for gaps between 2000 and 0000UTC. About 0200UTC the band opens to the US South- then Mid-West (upper quartile and mean very close), followed very soon by a step in the upper quartile from propagation extending to the Eastern Seaboard. About 1300UTC the path to the Eastern Seaboard closes. The hours until about 1800UTC (spanning sunrise) show more day-to-day variability: on 29 July propagation was open to Australia (11,000km), but not on 30 or 31 July, the path to South Africa (19,000km) opened each day, albeit very briefly on the 31st.

For KK6PR the overall pattern is similar, but the band starts to open to the Eastern Seaboard 2-3 hours

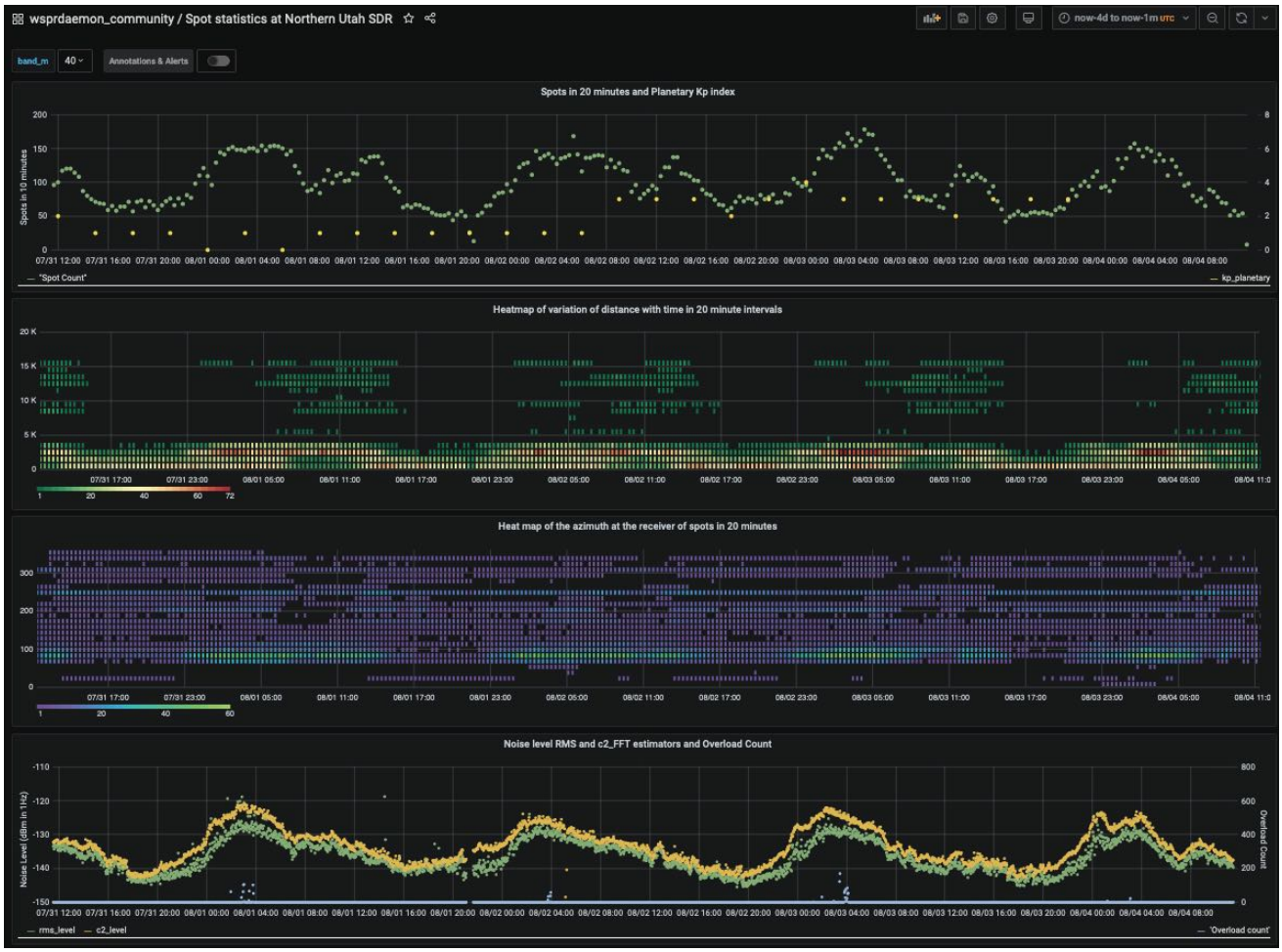

Figure 5. An example Grafana Dashboard that combines simple time series graphics from three sources with heatmap representation of data. The data spans four days for 40m from KA7OEI-1, a KiwiSDR at the Northern Utah SDR site. The top panel shows the number of spots received in 20-minute intervals in green, with three-hourly estimates of the planetary geomagnetic disturbance index Kp in yellow. The second panel is a heatmap of the number of spots in 20-minute intervals within 1000km distance bins out to 20,000km. The third panel shows azimuth at the receiver in 15° and 20 minute bins. The bottom panel shows the c2_FFT and RMS noise estimates with a distinct diurnal pattern.

earlier, at just before 0000UTC, and it is open for longer. The increased upper quartile distances spanning sunrise were from path openings to Japan, New Zealand and Australia.

While undoubtedly useful, these simple graphs require the user to consult listings of spots received to fully interpret the paths involved. Presenting the data as heat maps of distance and azimuth at the receiver helps (admittedly with ambiguity over short and long path).

5.2 Displaying other data and heat maps alongside WSPR spots

Figure 5 is an example Grafana Dashboard combining simple time series graphics from three sources with heat maps. The data spans four days for 40m from KA7OEI-1, a KiwiSDR at the Northern Utah SDR site [16]. The top panel shows the number of spots received in 20-minute intervals in green, with three-hourly estimates of the planetary geomagnetic disturbance index Kp in yellow. While there is day-to-day variation, there is a daily broad minimum spanning local noon. The heat map in the second panel shows a distinct daily periodicity in the number of statins received at a distance of 2500-4000km – from the Eastern Seaboard – an interpretation helped by the azimuth heat map in the third panel. However, reference to a list of received spots is still needed to check that the first evening DX is from South Africa, then New Zealand, followed by Australia and Europe. But on some days, southern European and North African stations appear during early evening. The bottom panel shows the c2_FFT and RMS noise estimates with a distinct diurnal pattern suggesting that, above the troughs, the noise was being propagated in rather than from a local source.

5.3 Visualizing derived data - SNR difference

As postgreSQL, the foundation of TimescaleDB, is a

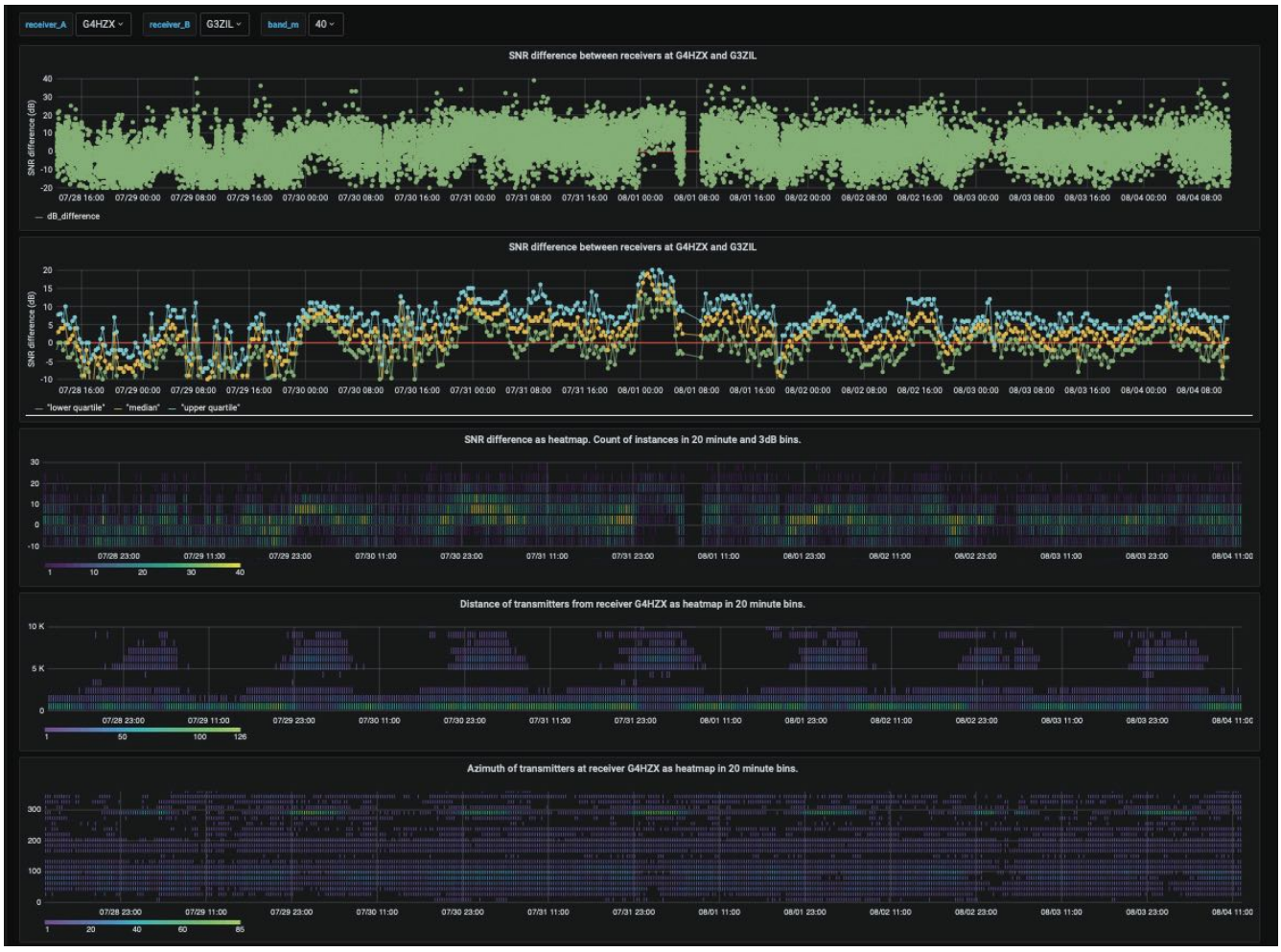

Figure 6. As postgreSQL is a relational database that allows self-joins a query in Grafana can return data derived from one or more receivers. This Dashboard shows the SNR difference between G3ZIL and G4HZX on 40m, for spots from the same sender at the same time, as a time series scatter plot in the top panel, as median and lower and upper quartiles in the second panel and as a heatmap in the third. Heat maps of spot distance and azimuth at G3ZIL help interpret the SNR difference.

relational database it is easy to use join queries, across different tables and databases but also within a table (a self-join). Grafana's query builder does not have sufficient options to construct a self-join but it is straightforward to write the necessary postgreSQL queries. Figure 6 shows a Dashboard designed to derive and plot the SNR difference between any two selected receivers where they spotted the same sender at the same time on the same band. The SNR difference is shown as a scatterplot, as median, lower and upper quartile in 20-minute intervals and as a heat map. Heat maps of the spot distance and azimuth at the first receiver are added to help interpretation.

In this example G3ZIL and G4HZX are 110km apart, G3ZIL generally has higher local noise than G4HZX, and the antenna is a vertical at G4HZX and a pair of low inverted V dipoles at G3ZIL. The statistics and heat map panels are the most useful for seeing the SNR differences. Clearly, a snapshot over a few minutes or even a few hours would not convey the complexity of the changes in SNR with time.

While there are undoubtedly short term variations there are clear diurnal (daily) variations as well as longer-term changes. Comparing the SNR difference and distance heat maps show higher SNR at G4HZX during hours of darkness with 40m open to the west, to North America, as seen in the azimuth heat map. The vertical out-performs the low dipoles for signals with low arrival angles. Conversely, spanning local noon, the low dipoles give a higher SNR than the vertical for signals from 0–1000km arriving at higher angles.

6. Conclusion

Implementing a database of our own has reinforced our opinion that wsprnet.org does a very good job at data collection and fulfilling simple web-based queries. With wsprnet.org as the primary data collector we have shown how a time series database and a visualization package can provide a richer set of data queries and graphical output.

Along the way we learnt about the strengths and weaknesses of time series databases. While Influx's documentation was excellent and a working system was easy to implement it took some time for its drawbacks in dealing with the very high cardinality of WSPR data to become obvious.

TimescaleDB's design better suits WSPR data, although on a machine with limited memory query response times into 10s of seconds occur for data not in the current time chunk. Our affordable solution has been to move to a server with 192GB of memory, sufficient to hold 30 days of WSPR data in memory, and with sufficient SSD and disk storage to provide (albeit slower, but still online) access to older data when required.

Grafana comes with built-in connection paths to TimescaleDB and other data sources. Its own visualization options for time series graphs and heat maps are an excellent starting point for examining WSPR spot data, as illustrated in this paper. Plug-ins from a growing community of third-party sources provide an even wider range of options such as forms of Hovmöller diagrams where the y-axis is hour of the day and the x axis time, and a range of maps.

Finally, there is clear potential from adding non-WSPR data. We have only scratched the surface in this regard, sourcing geomagnetic disturbance measurements from the US Space Weather Prediction Center. Through the means outlined in this paper we look forward to users making far more of the terrific resource that is the global WSPR community and its data.

Acknowledgment

We are grateful to Rick Whal (KK6PR), Clint Turner (KA7OEI) and Nigel Squibb (G4HZX) for permission to refer to their data, to constructive discussion and comment from WsprDaemon users at their weekly teleconference and to Glenn Elmore for his support, advice and indefatigable testing.

References

[1] github.com/garymcm/wsprnet_api and derivatives such as github.com/dl2sba/WsprNet
[2] Taylor, J. and Walker, B., 2010. WSPRing around the world. *QST*, *94*(11), 30-32.
[3] www.jimlill.com:8088/today_int.html
[4] wspr.pe1itr.com/
[5] mardie4.100webspace.net/wspr/
[6] wspr.aprsinfo.com/
[7] wspr.vk7jj.com/
[8] apps.apple.com/us/app/wspr-watch/id532487317
[9] wspr.fggs.de
[10] Robinett, R., 2019. WsprDaemon: A low cost, high performance, all band WSPR decoding system. *ARRL / TAPR Digital Communications Conference*, Detroit, 2019. www.youtube.com/watch?v=nHVN8oUUtlE
[11] Struckov, A., Yufa, S., Visheratin, A.A. and Nasonov, D., 2019. Evaluation of modern tools and techniques for storing time-series data. *Procedia Computer Science*, *156*, pp.19-28.
[12] docs.influxdata.com/influxdb/v1.8/introduction/get-started/
[13] blog.timescale.com/blog/what-is-high-cardinality-how-do-time-series-databases-influxdb-timescaledb-compare
[14] docs.timescale.com/latest/getting-started/installation
[15] github.com/timescale/outflux
[16] www.sdrutah.org/