



The Lasair-ZTF Transient Broker

Phase A prototype enabling analysis of the ZTF stream

Project Acronym LUSC-A
Project Title UK Involvement in the Large Synoptic Survey Telescope
Document Number LUSC-A-08

Submission date	1/5/2019
Version	
Status	Finished
Author(s) inc. institutional affiliation	Roy Williams, Ken Smith
Reviewer(s)	Bob Mann

Dissemination level	
Public	<i>Public</i>

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	04/04/2019	First version	Roy Williams, Ken Smith
1.0	28/05/2019	Reviewed and revised	Roy Williams, Ken Smith, Bob Mann

Table of Contents

VERSION HISTORY	1
1 EXECUTIVE SUMMARY	2
2 AN EVENT BROKER	3
3 PHASED DEVELOPMENT OF LASAIR	4
4 DESIGN OF LASAIR-ZTF	5
4.1 INGESTION	5
4.2 WEB AND SERVICE ASPECT	6
5 USER FUNCTIONALITY IN LASAIR-ZTF	6
5.1 STREAMS AND QUERIES.....	6
5.2 CONE SEARCH.....	9
5.3 COVERAGE	9
5.4 WATCHLISTS	10
5.5 SKYMAPS.....	11
5.6 INGESTION STATUS	12
5.7 JSON RESPONSES	12
5.8 JUPYTER	12
6 LESSONS LEARNED AND PHASE B DEVELOPMENT	12
7 GUIDE TO CODE	13
7.1 INGESTION	13
7.2 LASAIR WEBSITE	14
8 DATABASE SCHEMA	15
9 REFERENCES	15

1 Executive Summary

The Lasair broker in LSST Phase A is a prototype that ingests the ZTF sky survey [1] in preparation for ingesting the LSST sky survey. ZTF and its infrastructure has been designed for similarity with LSST, so infrastructure built, and lessons learned, in this prototype are an excellent preparation for the full LSST transient stream, that will be 10 to 100 times as voluminous as ZTF. In this prototype, we have a nightly ingestion process that has run for nearly a year without major failure, and we have released a public website that is attracting attention from the UK and international astronomers.

The philosophy of Lasair development is to evolve “from working to working”, so that we are starting with a broker for ZTF, developing its functionality on the basis of user experience, with scalability testing in parallel. So, this document provides a snapshot of the state of that development process at the end of Phase A and describes a broker being used to study the ZTF alert stream. This is a technical report on the current state of Lasair, not a user guide, and the user documentation is provided on the website itself.

2 An Event Broker

LSST will discover large numbers of astrophysical objects that move across the sky, change in brightness, or appear as transients. Because of their intrinsic temporal variability, full scientific exploitation of many classes of these events requires rapid discovery and additional follow-up observations. Accordingly, LSST's real-time Prompt Processing pipelines will identify such detections using image differencing and report them using world-public alerts issued within 60 seconds of the shutter closure after each visit. These alert packets will contain not only the information about the most recent detection but also a historical lightcurve, cutout images, timeseries features, and other contextual information. A science user will be able to use the contents of a single LSST alert packet to make a decision that is both rapid and informed about whether the event is relevant to their scientific goals. LSST expects to produce up to about ten million of these alerts nightly. The resulting alert stream will be large—more than 1 TB nightly—but will contain all classes of astrophysical events, from the youngest supernovae to the most distant RR Lyrae to the slowest-moving Trans-Neptunian Objects to the most unexpected new phenomena. The task for the scientist is to identify the small subset of events of interest in the larger alert stream. To do so, astronomers will rely on third party community brokers, software systems that receive the full LSST alert stream and provide additional information to refine the selection of events of interest. LSST will also provide an alert filtering service with more limited capability and capacity.

Community brokers may crossmatch the LSST stream to multiwavelength catalogs, join LSST alerts with those from other surveys, provide machine-learned classifications of events, and/or offer user filters to winnow the stream. LSST will itself offer an alert filtering service that will apply user-supplied filters to the alert stream. Individual scientists may receive alerts through one or more of these services. The large volume of the alert stream and the finite bandwidth from the LSST Data Facility necessitate a proposal process to select community brokers that will receive the full stream. This document outlines the process, criteria, and timeline by which LSST will choose community brokers. To provide context, we also summarize the major features of the LSST Alert Production systems as well as relevant data rights concerns.

Lasair will be a Community Broker for LSST. In preparation, the LSST-UK team has built a prototype, using phase A funding, to broker the ZTF[1] transient stream, that we call Lasair-ZTF. The objectives are to gain experience, to build and interact with a user community, to and to experiment with new technologies. In building Lasair-ZTF, we have adopted a user-centric strategy: asking for contributions from the community, then promoting the best of those to the system level, so they are available to all on the front page of Lasair. In this way, the most useful and best of that built by the community is made available to the entire Lasair community.

Spatial filters ask if the event is within a specified geometry (a subset of the celestial sphere), and include 'skymap' queries that come from gravitational-wave alerts, where the geometry is actually a probability density on the sky. They also include the 'multicone' type of geometry that consists of a set of sources (a catalogue), each with a radius, also called a watchlist. In the future, we expect to collaborate with the Virtual Observatory stakeholders to allow easy creation and exchange of both geometric and multicone filters.

Lasair also supports *annotation* of events, where new information is added to the event to form an 'event portfolio'. We refer to 'new' events as those that come directly from the LSST source, and 'rich' events as those that have a portfolio. Annotation can be created, for example, by contextual search of massive catalogues, where nearby 2MASS or PanStarrs sources are found and associated with the event. Lasair also computes statistics of the light curve or each object, such as minimum and maximum magnitude,

etc. Other annotations include user comments on objects, and crossmatches with the Transient Name Server (TNS)[3].

In the future, we will encourage annotation from users: by running their code on the light curve or stamp image, and adding a report packet to the original event. Codes for building annotation will be run either in batch mode (for historical events), or in real-time mode, as part of the ingestion pipeline. Some of the filters that decide the importance of events will only need to run on new events, and some will need a combination of parts from the portfolio.

When a filter is built using only the new event, it can run more quickly. In the Kafka streaming architecture, the stream of new events can be replicated and thus scaled over many processors, and complex queries executed in parallel. The new event contains rich information, including the full light curve of the source associated with the event.

A future capability of Lasair will allow filters to run in real time, as part of the ingestion pipeline, its output may result in sending an alert message to a user. Such alerts will take many forms (text, VOEvent, GCN Notice, etc), and will be delivered in many ways (email, SMS, Slack etc). Lasair is investigating different ways to make this variety possible for users and manageable by the project.

Lasair offers a number of services already to give access to the event archives. Cone search is a local spatial filter based on a single position and radius, and a watchlist filter is one from a spatial filter that is the union of many cones. A filter can also be built from an SQL SELECT query, so long the output is both time-ordered, and includes either a new or rich object. Coverage service shows the sky where the survey has covered, according to which dates and telescope filters have been used. The ingestion status service shows current status of the ingestion of events from the telescope.

Lasair uses the Sherlock [4] system to give context to the candidates ingested from ZTF. Given a position in the sky, Sherlock determines likely known objects that it belongs to. A candidate in a galaxy is very likely to be a supernova, so the context information is crucial. Users can query on the Sherlock context information to select their objects of interest.

3 Phased development of Lasair

in parallel with this working prototype for the ZTF stream, we are also conducting scalability experiments with alternative technologies, some of which are likely to be adopted in future versions of Lasair, so that we end up with a system that meets the scalability requirements for the LSST stream. Users will still interact with the events by providing filters based on sky geometry and SQL, but their use and scalability will evolve. The geometric filter will include not just watchlists of sources, but also general sky areas. The SQL SELECT filter will be not only for user-initiated queries (“click here to submit query”), but also automatically to generate a substream of the original event stream, that can be stored, disseminated, or used for alerting humans. Filters will be a combination of geometry and SQL. Filters will be able to utilise simple code as UDFs (User Defined Functions), and complex assessment of the event through annotations: searching catalogues, light-curve analysis, etc. Queries will not run until the annotations they need are available.

Finally, of course, all of this will be developed in a scalable way, using Apache Spark for parallelism over a cluster of “shared nothing” nodes that partition the sky. We also expect to replace the MySQL database with a more scalable data storage, but retain the SQL language as far as possible, since many scientists are already familiar with it.

4 Design of Lasair-ZTF

Lasair-ZTF uses a technology stack: Kafka, Django, MySQL, Jupyter.

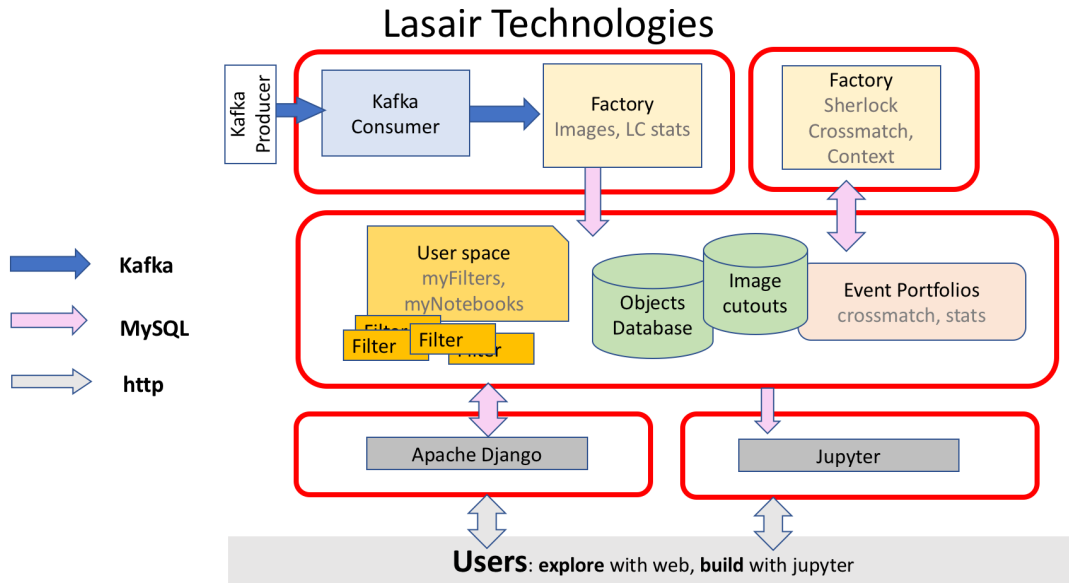


Figure 1: Lasair-ZTF Architecture, May 2019: Five hosts drive Lasair, four are VMs on one machine (Isstuk2), and the jupyter interface is on a different machine (Isstuk3). This system has been reliably ingesting the ZTF stream for about a year. Events arrive from a Kafka Producer to Isstuks2:lasair-head (top left), and go into a relational database, with the image stamps in the file system (green). The Sherlock code[4] has been developed at QUB for many years to evaluate local context and infer event properties; it runs in the background on Isstuk2:lasair-node0 (top right). Other Factory tasks are: converting FITS to Jpeg, building a coverage table to show on the website, summary statistics of the light curve. The MySQL data base on Isstuk2:lasair-db saves the candidates and objects, as well as data generated from them ("portfolio"). Users can store their own information (yellow), including "watchlists" of sources, saved SQL queries, and Jupyter notebooks. These run on demand by the user, but cannot yet be put into the real-time pipeline. There are no alerts being pushed, rather everything is through the web browser and services, and Jupyter.

4.1 Ingestion

The stream of transients is ingested from the ZTF survey from the University of Washington in the USA (`public.alerts.uw.edu`), using the Kafka software, that is designed for high-throughput streaming applications. The stream consists of "candidates", each referring to a detection in the sky of a source that is different from the reference images, that were taken in early 2017. Each candidate has numerous attributes [9], as well as FITS images of the detections. Each candidate is immediately inserted into the MySQL database, and the image files into the file system. As of May 2019, the database has 35 million such, and each clear night delivers another ~200,000.

Data is fetched from ZTF in batches of 50,000, after which the streaming stops, followed by further ingestion processes, such as converting the image files to Jpeg for web display, collecting together all detections of the same astrophysical object, and characterising that light curve, and updating the coverage information for the survey. After a 10-minute wait, the system polls for new candidates, and the cycle starts again. The node with Sherlock scavenges for objects in the database not yet characterised, and works on those.

4.2 Web and service aspect

Lasair-ZTF is a combination of Kafka ingestion, a web server, and a Jupyter server. The web aspect is built with Django, a web-application framework where each URL drives python code; the database is seen by this code as a persistent object store, and results from the code are passed to templates that generate the output web pages. Users can sign up with a self-service system, that allows them to save their queries, to add comments to interesting objects, and to save watchlists of interesting sources. Users can define an arbitrary SQL SELECT query, which is then modified with a timeout (max execution time directive), and a limit of 1000 returned records. Once this first page of 1000 is delivered, a 'next page' button fetches the next group.

The Lasair web server makes good use of AladinLite[7], that provides a zoomable sky background built from a wide selection of image and catalog products, including PanStarrs and Gaia. This display is used to show the neighbourhood of an specified object, to show coverage of the ZTF survey, and to show gravitational skymaps. The Lasair-ZTF web also makes use of Plotly for zoomable light curves, and Bootstrap styling to make the web pages useful for a wide variety of devices.

5 User functionality in Lasair-ZTF

Lasair is a website, usable by humans and by machines, and it is a Jupyter notebook hub. The services offered are listed in the left margin of

<https://lasair.roe.ac.uk>

and they are described below. There is a linked effort to develop a Jupyter notebook interface to the transient stream and its products.

5.1 Streams and Queries

This page is a SQL-builder for the ZTF database, and is in two parts. The primary page ("Lasair Streams") lets a user run some SQL without actually seeing it -- the streams page -- as many people are afraid of SQL. The SQL itself is on a secondary page, ("click here to build your own query"), and On the streams page is a big box where the user can type in raw SQL. Alternatively, they can click on a query below, which fills the big box with the corresponding SQL. There are three kinds of saved queries:

- My queries
- Contributed (public) queries
- Lasair system queries

The first is presented as a form interface, so that the owner can change the query, its name, and its description. There is a check box labelled "public" meaning that others can see your query. The Lasair system query is a subclass of Public query, and is shown on the streams page. Currently these favoured streams are:

Name	Description
SN-like candidates in last 14 days	SN-like candidates (Sherlock classifications SN, NT and orphans), time limit adjustable (just adjust the number 14). Rejects Pan-STARRS star matches

LASAIR EVENT BROKER

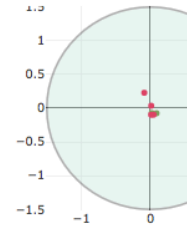
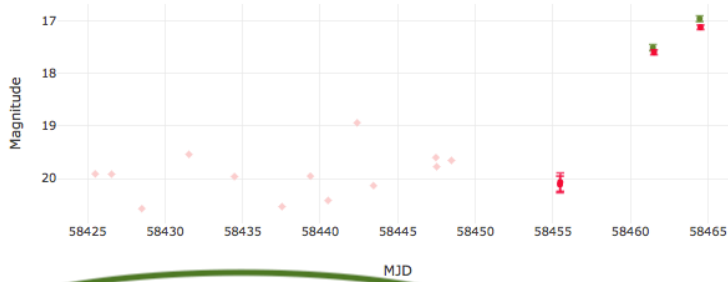
Name	Description
<u>All nuclear transients and TDE candidates</u>	Near core of inactive catalogued galaxies (within 1"), flags Pan-STARRS stellar matches to let user judge star/galaxy separation. Objects discovered in last 30 days.
<u>TNS crossmatch</u>	This query finds all Lasair objects that are in the <u>Transient Name Server</u> [3], meaning they have a comment that includes the string 'TNS'. The most recent are first.

A user-built SQL query must begin with SELECT, so that it can't delete or alter the data, and a LIMIT 1000 is added to the query, to manage the size of the output. There is also a timeout applied to the query. The query is built from the following tables:

- **Candidates:** the detections ingested from ZTF, the same schema [6]. The primary key is `candid`.
 - **Noncandidates:** a table of nondetections at the places where there have been detections
- **Objects:** A collection of detections, all at the same place in the sky, within 1.5", the primary key is `objectID`.
- **Sherlock_crossmatches:** Objects from published catalogues that are close to a given object.
- **Sherlock_classifications:** The classifications of nearby (and coincident) objects from published catalogs The classification can be AGN, BS, CV, NT, ORPHAN, SN, VS, as well as UNCLEAR, or NULL.
- **Comments:** Users comments on objects, includes Lasair Bot who comments if the object is in the Transient Names Server [3]

Once a query is executed, the selected objects can be viewed in detail. Note on security: Allowing people on the internet to execute code on your server is always a tricky proposition. But it is what we have done for Lasair, in this case SQL code. It is run as a separate SQL user account that has restricted privilege, the tables that can be accessed carefully defined. Only a single SQL statement is allowed, and it must be a SELECT. Before execution, the query is modified to add a limit to the output and a timeout.

Object ZTF18acsovsw



Roy Williams Dec. 17, 2018, 10:30 a.m. [TNS 2018jny](#)

Comments from others

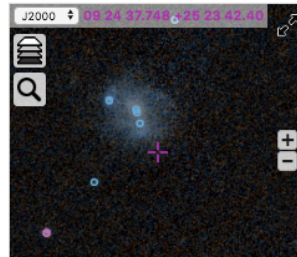
Crossmatches

rank	ID	Catalog	Type	Separation	r-mag	g-mag
1	1237667143404486939/1735077/SD	SDSS/GLADE/NED	galaxy	11.65	17.1979	17.6802

Host galaxy

AladinLite

- DSS color
- SDSS color
- 2MASS
- AllWISE
- GALEX



PanSTARRS view with sources

Figure2 : Rich Object Information: The light curve of each object is shown in two filters (g and r), with both detections and non-detections, in this case for supernova 2018jny / ZTF18acsovsw. Comments can be added by logged-in users, such as this one with the link to the IAU name of the supernova. The Sherlock classification system has provided information about the host galaxy. Aladin Lite[7] provides both images and catalogue sources: here we see Pan-STARRS image of the host galaxy with sources from Pan-STARRS and Gaia DR2.

LASAIR EVENT BROKER

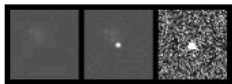
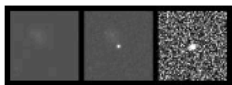
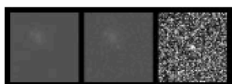
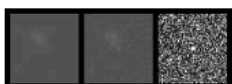
UTC	Filter	magpsf	candidate	Image
2018-12-12 12:34:57	r	17.118 ± 0.041	710524282815015057	
2018-12-12 11:14:45	g	16.956 ± 0.055	710468572815015056	
2018-12-09 12:33:08	r	17.595 ± 0.046	707523002815015072	
2018-12-09 10:54:16	g	17.505 ± 0.059	707454352815015051	
2018-12-03 11:31:49	r	20.066 ± 0.176	701480432815015064	
2018-12-03 11:06:05	r	20.110 ± 0.157	701462552815015044	
2018-11-26 11:04:25	r	19.653	non-detection	
2018-11-25 12:23:26	r	19.773	non-detection	
2018-11-25 11:05:13	r	19.599	non-detection	

Figure3 : Detection Images: The object page has links to detailed information about each observation that makes up the light curve, with images, in this case for ZTF18acsovsw. We see the brightening of 2018jny.

5.2 Cone Search

The cone search facility is available from every Lasair page in the top bar. An expression of right ascension and declination in many forms can be entered here, in decimal degrees or sexagesimal, with many of the usual delimiters. In this way, a sky position can be copy/pasted directly from some other place. An optional cone radius can also be added last, as a number of arcseconds. Alternatively an objectID (eg ZTF18aaalymq), or a comma-separated sequence of such can be entered.

5.3 Coverage

ZTF uses a fixed set of *fields* for its observations, laid out in a RA/Dec grid on the sky. The coverage page of Lasair shows which of those fields have been covered by the ZTF survey, in a given date range, with red squares for r-band observations and green squares for g-band. The AladinLite [7] image can be manipulated to show more detail and more quantitative information. Below the image, coverage is given in text form, being

the number of candidates returned from each of the fields.

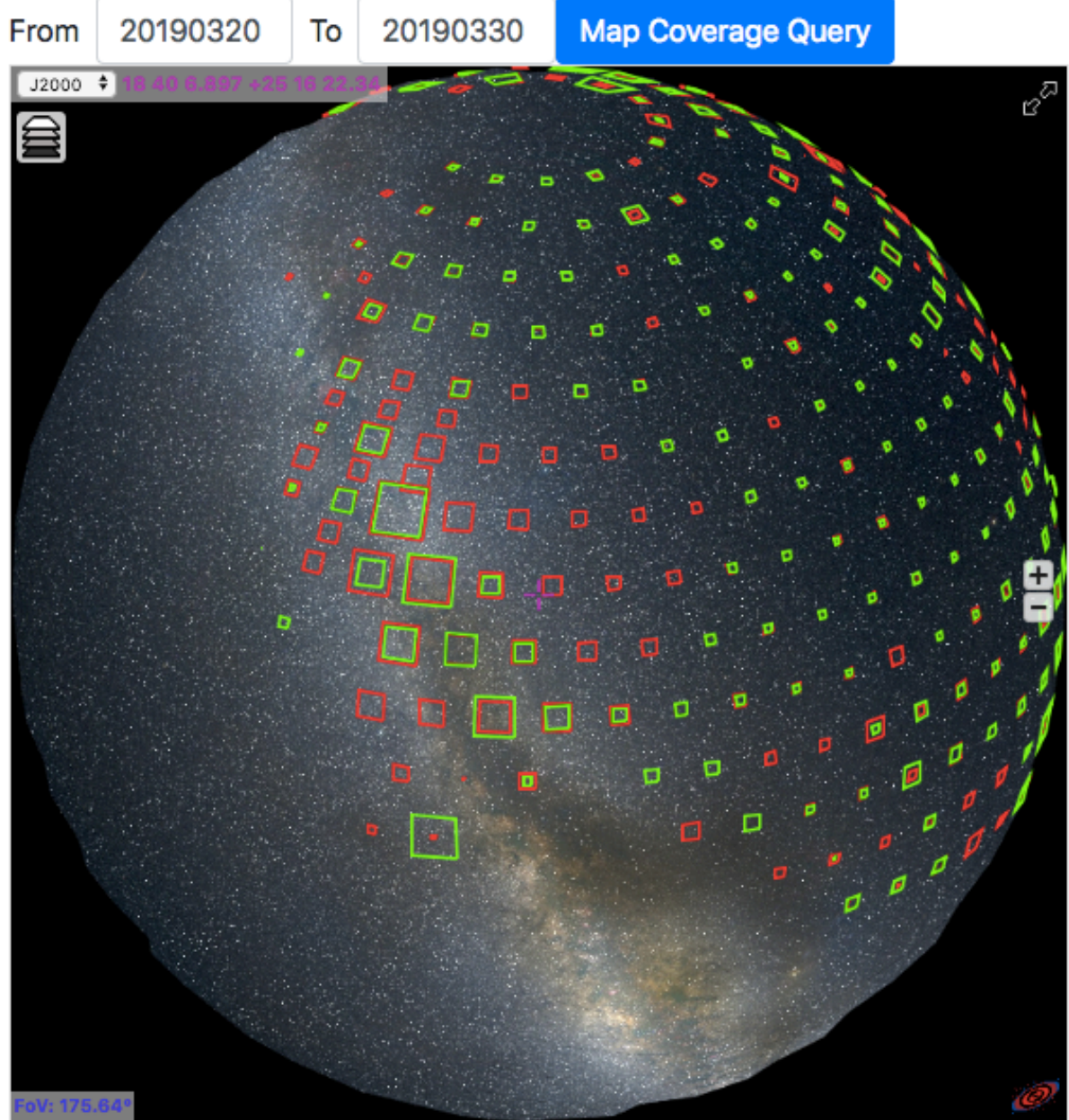


Figure 4: ZTF coverage.

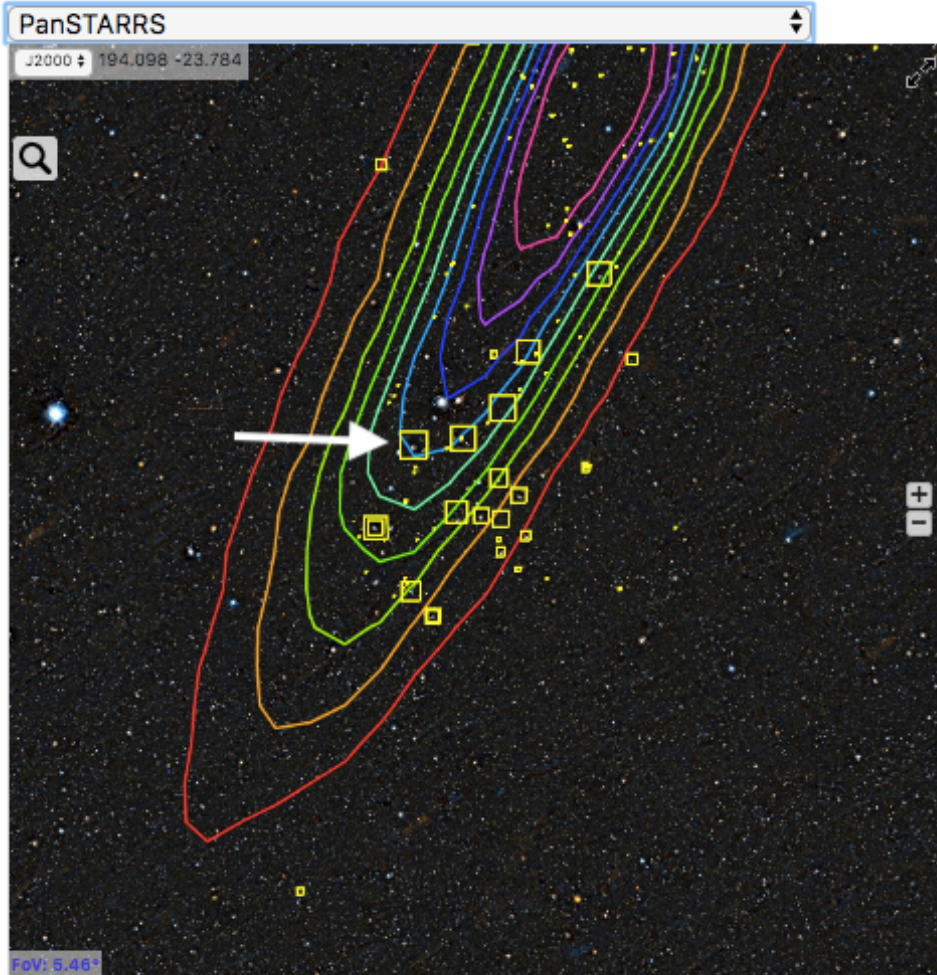
5.4 Watchlists

A watchlist is a set of named points in the sky, with a radius in arcseconds. Users can input a watchlist and save it, and they can run a crossmatch between their watchlist and the ZTF database. For example, if an astronomer is interested in “*BL Lac candidates for TeV observations*” [8], they can build a watchlist with these, to look for optical transients from ZTF that come from these special galaxies. Watchlists can be public or private. The result of the crossmatch is a table with the left columns being the watchlist sources, and the right columns the corresponding ZTF object, with the number of candidates shown, and the Sherlock classification.

Each watchlist also has a checkbox “active”, which currently has no effect. In the future, however, we expect this to mean that the crossmatch is run as soon as possible after candidates are ingested, with alerts sent to the user to say that a transient has appeared on one of the watched sources.

5.5 Skymaps

Coverage of ZTF From To
 Show ZTF candidates From To
 Show galaxies from [GLADE](#) *doubleclick a galaxy*



Galaxies with more than 1% probability:

Percent	Link to NED	Distance (Mpc)
9.11	NED	39.35 ←
8.00	NED	38.03
7.15	NED	43.15

Figure 5: Lasair is connected to the LIGO-Virgo gravitational wave observatories, and receives and processes fresh observations within minutes. The result is a page such as that shown here, which corresponds to the “golden event” called GW170817. The contours of probability are shown for the skymap, in 9 levels from 10%, the innermost contour, to 90%, the outermost. Controls above allow the coverage of the ZTF survey to be shown, as well as recent ZTF transients that may be the counterpart of the GW observation. The yellow squares in the image surround likely galaxies where the counterpart might be found, where the area of the square is proportional to the probability it contains the counterpart. These probabilities are

listed in text form below. In the case of GW170817, the counterpart was found in NGC 4993 (white arrow), which is shown with a ~10% chance (black arrow).

5.6 Ingestion Status

There are three processes running frequently on the Lasair ingestion machine:

- Kafka ingestion from ZTF
- Crossmatch with Transient Name Server
- LIGO-Virgo skymap ingestion

Each of these produces a log file, that can be seen from the Lasair Ingestion Status page. The first of these shows the time of the most recent update, the number of minutes since then, the number of alerts that ZTF is reporting for the day, and the number that Lasair has ingested – which should be quite close.

5.7 JSON responses

Several of the web pages described above can return a JSON response, made for machine use, in place of the HTML/JS that is made for human understanding. These are:

- `/conesearch/`: see check box for JSON output.
- `/objlist/`: when submitting an SQL query, see check box for JSON output
- `/object/<objectId>`: for viewing a specific object, there is a JSON output
- `/skymap/<skymapId>`: for viewing a specific skymap there is a JSON version.

5.8 Jupyter

There is also a prototype Jupyterhub installation (on Isstuk3) that can make a connection to the Lasair database, using the `mysql.connector` package. There was a meeting of UK astronomers [ref] interested in using Lasair, and many built Jupyter notebooks. There are many examples at <https://lasair.roe.ac.uk/jupyter>.

6 Lessons learned and Phase B development

Lasair-ZTF has been a success, providing astronomers with meaningful access to the ZTF transients since May 2018, with very little downtime. It has relied on Kafka for ingestion, MySQL for database storage, Apache/Django for web aspect, and provided a Jupyter aspect that can see the database. Users provide geometric criteria in the form of watchlists, and SQL SELECT queries to pick interesting events, that can be followed up in detail via Jupyter.

In Phase B we expect to achieve the following:

- Allowing users to create more complex queries, combining watchlist, geometry, and SQL
- Addition of more annotations in addition to Sherlock, for example a machine-learning code to classify light curves.
- Converting SQL SELECT queries into filters that generate substreams of interesting events.
- Running these filters along with event ingestion to generate streams in real time. In particular, a filter will only run when the annotations it needs have been computed.
- Combining all the above to build a supernova stream that 'mostly' contains supernovae, and contains 'most' supernovae that ZTF has seen.

- Delivering event substreams to both end-users and to downstream processing facilities.
- Experimenting with LSST commissioning data and porting from ZTF to LSST.
- Building parallel and multi-threaded systems, together with scaling experiments to increase and optimise the flow of data through the many moving parts of Lasair.

7 Guide to Code

The following is a guide to the code that drives Lasair. The whole code is public and available from a github tag dated 28 May 2019

<https://github.com/lsst-uk/lasair/tree/lasair-ztf>

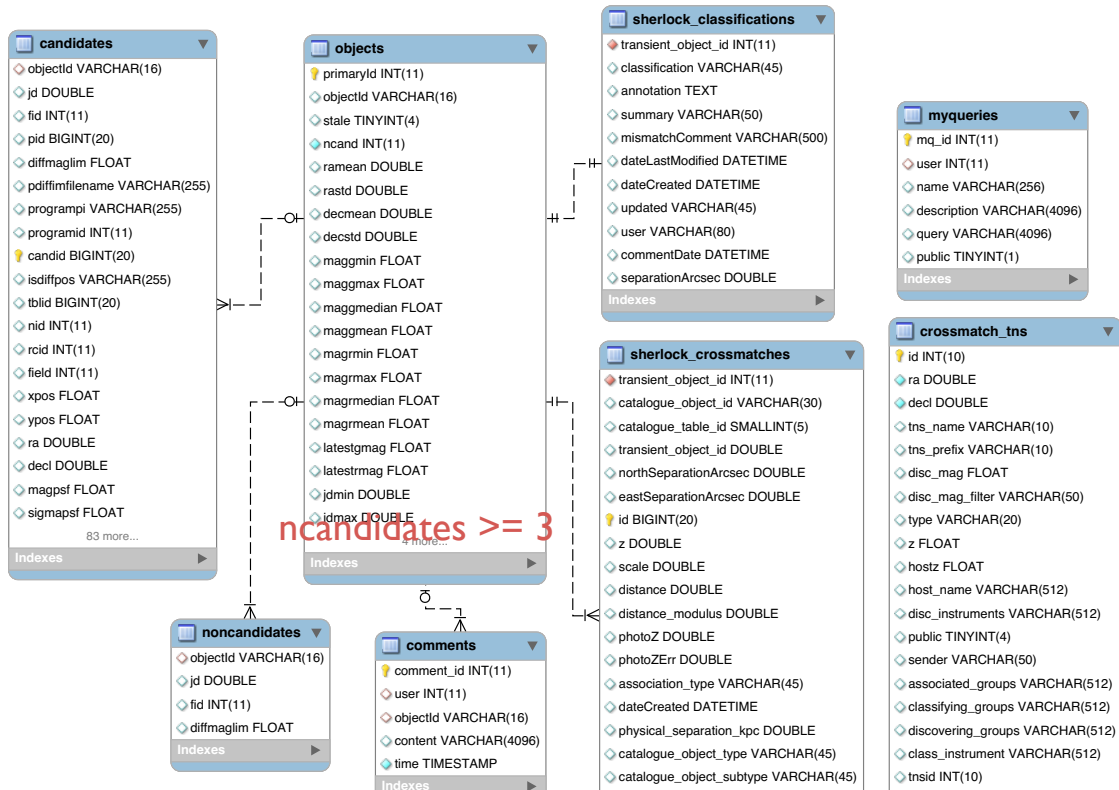
7.1 Ingestion

- **alert_stream_ztf:**
 - **bin/ingestStream.py:** the Kafka consumer.
 - **alert_stream_ztf/common**
 - **date_nid.py:** converts night-id integer to/from normal dates
 - **htm:** the Hierarchical Triangular mesh code for crossmatches is used in the watchlists, the cone search, and the TNS crossmatch.
 - **run_crossmatch.py:** Code that runs when a user runs the watchlist crossmatch from the web page
 - **run_tns_crossmatch.py:** Code to crossmatch TNS with ZTF
 - **settings.py:** Secrets that are not in the github repository. Database passwords etc.
 - **python/lst/alert/stream**
 - **alertConsumer.py:** Imported by ingestStream.py, this is the Kafka consumer implemented with Confluent Kafka.
 - **ztf_ingest.py:** The code that runs every 10 minutes to poll Kafka, get the latest candidates, then run the Factory tasks in post_ingest.
 - **ztf_ingest_log.py:** Wrapper around ztf_ingest.py that runs and waits, runs and waits. Also puts all output from ztf_ingest.py into a log file.
- **post_ingest:**
 - **check_status.py:** Builds the status summary that is visible on the ingestion page.
 - **coverage.py:** Builds the coverage table that is used by the coverage page.
 - **get_number_candidates.py:** Computes the number of candidates in the database so that it can be visible on the “about” page.
 - **jpg_stamps.py:** Looks for triples of FITS stamps that do not have a Jpeg version, and makes the Jpeg version, for the object page.
 - **poll_tns.py:** Fetches all the transients from the Transient Name Server (<https://wis-tns.weizmann.ac.il/>), and puts them in a database table.
 - **update_objects.py:** Rebuilds summary statistics for each object. Once a candidate is added to an object, the object becomes stale, and this code freshens the objects.
- **skymap:**
 - **lvc_gcn_listener.py:** Fetches LIGO-Virgo skymaps immediately it is published.
 - **skymapInfo.py:** Builds a JSON version of the skymap file, with contour lines, likely galaxies, etc.
 - **galaxyCatalog.py:** Part of skymapInfo.py.

7.2 Lasair website

- **./lasair-webapp/lasair/lasair:**
 - **candidates.py:** Showing a specific candidate from the database
 - **comments.py:** User comments on objects
 - **models.py:** Python classes for the Django models of the database tables.
 - **myqueries.py:** Stored queries and query execution.
 - **objects.py:** Making a page for a given object.
 - **services.py:** Services called by AJAX from Javascript, including coverage.
 - **settings.py:** Django settings, includes, secrets, not in github.
 - **skymap.py:** Building a page about a LIGO-Virgo skymap.
 - **urls.py:** The Django URLs and which code runs from which URL.
 - **views.py:** Miscellaneous web functions.
 - **watchlists.py:** Dealing with Watchlists.
- **./lasair-webapp/lasair/lasair/static**
 - admin, images, CSS, static content, includes AladinLite, JQuery, Bootstrap, Plotly.
- **./lasair-webapp/lasair/lasair/static/js**
 - **coverage.js:** Builds the red and green rectangles on the sky for the coverage page
 - **lc.js and lc_apparent.js:** Build the light curve display on the object page, one for difference magnitudes, the other for apparent magnitudes.
 - **skymap.js:** Build contours, galaxies, ZTF coverage, and ZTF transients for the skymap page
 - **surveys.js:** List of HiPS surveys to include for the AladinLite pages.
- **./lasair-webapp/lasair/lasair/templates**
 - **about.html:** /about page
 - **base.html:** The heading and left margin, included in all other templates
 - **cand.html:** Single candidate page
 - **candlist.html:** List of candidates page
 - **conesearch.html:** Conesearch page
 - **contact.html:** Contact page
 - **coverage.html:** Coverage page
 - **error.html:** Error page
 - **index.html:** Front page of Lasair
 - **jupyter.html;** Jupyter page
 - **new_comment.html:** To add a new comment to an object
 - **new_myquery.html:** To add a new stored query
 - **objlist.html:** Showing the result of a user-initiated SQL query
 - **objlistquery.html:** Inviting the user to make a SQL query
 - **release.html:** Release notes
 - **schema.html:** Tables and attributes to go into user-initiated SQL queries
 - **show_myquery.html:** Showing a specific stored query
 - **show_object.html:** Showing an object
 - **show_skymap.html:** Showing a LIGO-Virgo Skymap
 - **show_watchlist.html:** Showing a watchlist
 - **signup.html:** Signing up for an account
 - **skymap.html;** List of available skymaps
 - **status.html:** Ingestion status page
 - **streams.html:** Showing special queries without any SQL
 - **watchlists_home.html:** Watchlists page
- **./lasair-webapp/lasair/lasair/templates/registration:** Everything in this directory is about the self-signup procedure, logging in and out, forgot my password, etc.

8 Database schema



9 References

- [1] Zwicky Transient Facility, <https://www.ztf.caltech.edu/>
- [2] Lasair Transient Broker, <https://lasair.roe.ac.uk/>
- [3] Transient Name Server: <https://wis-tns.weizmann.ac.il/>
- [4] Sherlock: A python package with command-line tools for contextually classifying variable/transient astronomical sources. <https://qub-sherlock.readthedocs.io/en/stable/>
- [5] Lasair Astronomer's Meeting 9th October 2018 <https://lsst-uk.atlassian.net/wiki/spaces/LUSCSWG/pages/583237641/LASAIR+astronomer+meeting+9th+October+2018>
- [6] ZTF Avro Schemas <https://zwickytransientfacility.github.io/ztf-avro-alert/schema.html>
- [7] AladinLite interface: <https://aladin.u-strasbg.fr/AladinLite/>
- [8] F. Massaro et al, BL Lac Candidates for TeV Observations <https://ui.adsabs.harvard.edu/?#abs/2013ApJS..207...16M>
- [9] ZTF schema <https://zwickytransientfacility.github.io/ztf-avro-alert/schema.html>
- [10] Github tage for Lasair-ZTF <https://github.com/lsst-uk/lasair/tree/lasair-ztf>