Theme 7 – Group 2 Data mining technologies
Catalogues crossmatching on distributed database
and application on MWA absorption source finding

Crossmatching is a method to find corresponding objects in different datasets. The goal is to collect an object's observation in multi-wavelength or even multi-messengers, then astronomers could research more about the object. A simple way to crossmatch is calculating two objects distance on sphere via geometrical formula. But it takes a lot of time and not fit for the big data.

Catalogs are usually maintained in databases, which is well constructed, high performance and easy to query. Lots of query engines are built on MySQL, PostgreSQL, SQL Server et al. Crossmatching is also working on database, e.g. the SDSS project crossmatch their catalogs with WISE and other catalogs on SQL Server.

A. Distributed databases

Traditional relational database can only work on single machine which is limited by the hardware and bandwidth. In the big data ear, especially the SKA era, we need better solution to enable more calculation power to do data query and crossmatching. Furthermore, it would be excellent that if we even do not have to change our work habit. With the development of cloud computing, some possible solutions is coming out: the distributed database. Distributed databases are running on multi-node hardware.

During this workshop, we simply compared several distributed databases: GreenPlum, Hive and SparkSQL. As table 1 shows, they are all relational DBMS, supports data partition, and SQL friendly.

	GreenPlum	Hive	SparkSQL
SQL Support	Yes	HiveQL	Queries similar to SQL syntax
Triggers	Yes	No	No
Partitioning methods	Sharding	Sharding	yes, utilizing Spark Core
Database model	Relational DBMS	Relational DBMS	Relational DBMS
Spherical Geometry	PostGIS		

Table 1, simple comparison of several distributed databases

Since the time is quite limited. We chose GreenPlum which supports SQL to test the mature crossmatching method: Zones Algorithm. Zones Algorithm is invented by Jim Gray, a great computer scientist from Microsoft Research. The algorithm is well working on SDSS project, on their SQL Server database. Other crossmatch engine/scheme, such as HEALPix, HTM, Q3C, pgsphere are all too complicated, as a plug-in to enhance the specified database. The pure SQL implement of Zones Algorithm is more simple and flexible to run on much more databases.

B. Zones Algorithm

Zones Algorithm divides the celestial sphere to equally height zones, or call it rings, as Fig. 1 shows. Each zone has a unique ID, ZoneID=$\left\lfloor \frac{dec+90°}{zoneheight} \right\rfloor$. For convenience, the zoneheight is always a little larger than the crossmathing threshold, e.g. threshold=2”, zoneheight=2.1”. There are three main filter existing in the Zones Algorithm. The zones filter, declination filter and alpha filter.

Zones is the rough filter to skip most of the unrelated objects: any object "b" matches object "a" exists only in the 3 nearby zones of "a", $ZoneID_a - 1 \le ZoneID_b \le ZoneID_a + 1$.

Declination filter is more obvious, if abs(dec$_b$-dec$_a$)>threshold, object "b" will not be the corresponding object of "a".

Alpha filter is the most complicated part. ADD/SUBSTRACT or LAGER/LESSER comparison is easy and fast on calculation, but we cannot do it one Right Ascension in this way: $RA_a - threshold \le RA_b \le RA_a + threshold$. Threshold in each zone corresponding to different RA rangle, we call it "Alpha". But it is not impossible to do that. With some sphere geometry deduction, we have an equation to calculate the alpha: $\texttt{Alpha} = \left|\arctan\left(\frac{\sin\theta}{\sqrt{\cos(\delta-\theta)\cos(\delta+\theta)}}\right)\right|$, where δ is declination, θ is threshold. Now, we can simply filter RA by: $RA_a - Alpha_\delta \le RA_b \le RA_a + Alpha_\delta$.

Fig. 1. Zones Algorithm cuts the celestial sphere to lots of equally height zones

There are some other problems we have to pay attention, e.g. RA=0 or 360 warp around, duplicate matching. Much more details could be found in Gray et al.(2007).

C. Test and comparison

We test the Zones Algorithm on GreenPlum[1] with three catalogue:

- CSTAR(Chinese Small Telescope Array), which locates at Antarctic Dome A, the catalog contains 20,000+ rows of objects.
- LAMOST DR2, has 4,000,000+ rows.
- TwoMass Point Source Catalog, 470,000,000+ rows.

The threshold is 2", and the zoneheight is 2.1". We also run similar code on a PostgreSQL[2] database to compare the performance. As Table 2 shows, on small data, PostgreSQL is faster than GreenPlum, but LAMOST DR2 GreenPlum is better than PostgreSQL. TwoMass PSC catalog is too large, we cannot get the result during the workshop.

	CSTAR self	LAMOST DR2 self	TwoMass PSC self-matching
GreenPlum	10003ms	1017730ms	14350370ms+ not completed

[1] GreenPlum runs on two group resources on Alibaba Cloud: 4Cores / 32GB RAM /2TB HDD / 400 MBPS; 8Cores / 64GB RAM /4TB HDD / 800 MBPS

[2] PostgreSQSL runs on a physical machine: 32 Cores/ 132GB RAM / CentOS 7

PostgreSQL	9364ms	3120511ms	35522160ms+ not completed

Table 2. Performance comparison between PostgreSQL and GreenPlum

The GreenPlum works in this simple test, but it is not shining enough. The data might not be distributed/partitioned in suitable way. During our test, we found the data are partitioned by their zoneid, e.g. [1,3,5,7,9] in partition 1 while [2,4,6,8,10] in partition 2, which will lead to too much I/O cost. We should much deep into the database, choose better index to partition the data. More tests is also needed, including some attempt on other databases.

D. Extended source finding

During the cross-matching test on GreenPlum, we also tried to apply the method to improve the extended source finding and latter processing. Source finding is essential for data mining. The existing packages have a high performance on point source detections. However, very few packages can do extended source finding well due to these sources have complex structures or low brightness. Nearly no packages can detect the absorption features so far.

The future SKA survey will have a significantly high angular resolution, which will resolve much more radio galaxies, HII regions, supernova remnants, etc. Thus, we need to improve the current algorithm of extended source finding packages.

Our goal is automatically finding the extended sources in fits images. Two challenges exist for source detections. First, some sources only show absorption features in images. Second, faint sources are mixed with their surroundings which can not be detected. Fig. 2 shows the outline of data flow.

Aegean is great package for point source detections. We basically use Aegean to do the source finding and improve the accuracy using the following methods.

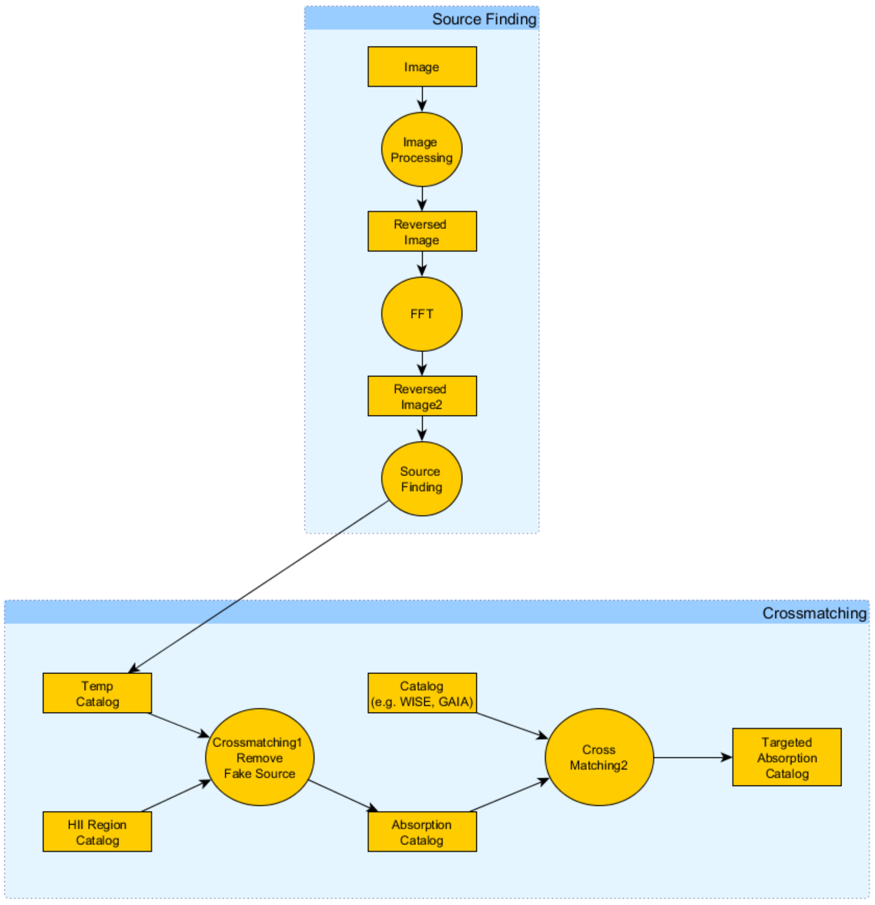

Fig.2. Data flow outline of the absorption source finding.

E. Method for source finding

We have found four solutions for the above challenges. First, we revert the image to make the absorption features become the emission features and then do source finding. To detect the absorption features, we simply revert our image using the maximum pixel value of the image minus each pixel value. This makes the absorption features appear to be emission features.

Second, we scale the fits image to make the faint sources appear to be bright sources for detecting them. We replace each pixel value with their exponential value. This make the source region appears to be bright and much easier to find them in Aegean.

Third, we use cross-matching to remove the false detections. The absorption features we detected show emission features at other frequencies. By cross-matching the detected sources at different frequencies, we can significantly remove false detections.

Last, we use the FFT method to remove the sources which have different angular scales with our target sources. This method help to remove the background and the unrelated sources or fake features. We use Fast Fourier Transform (FFT) to transfer our image to UV map. Then we select a certain UV ranges that only resolve our target sources with a certain angular scale. Then we do the inverse FFT to get an image with only target sources in it. Then we perform source finding using Aegean.

F. Results of source finding

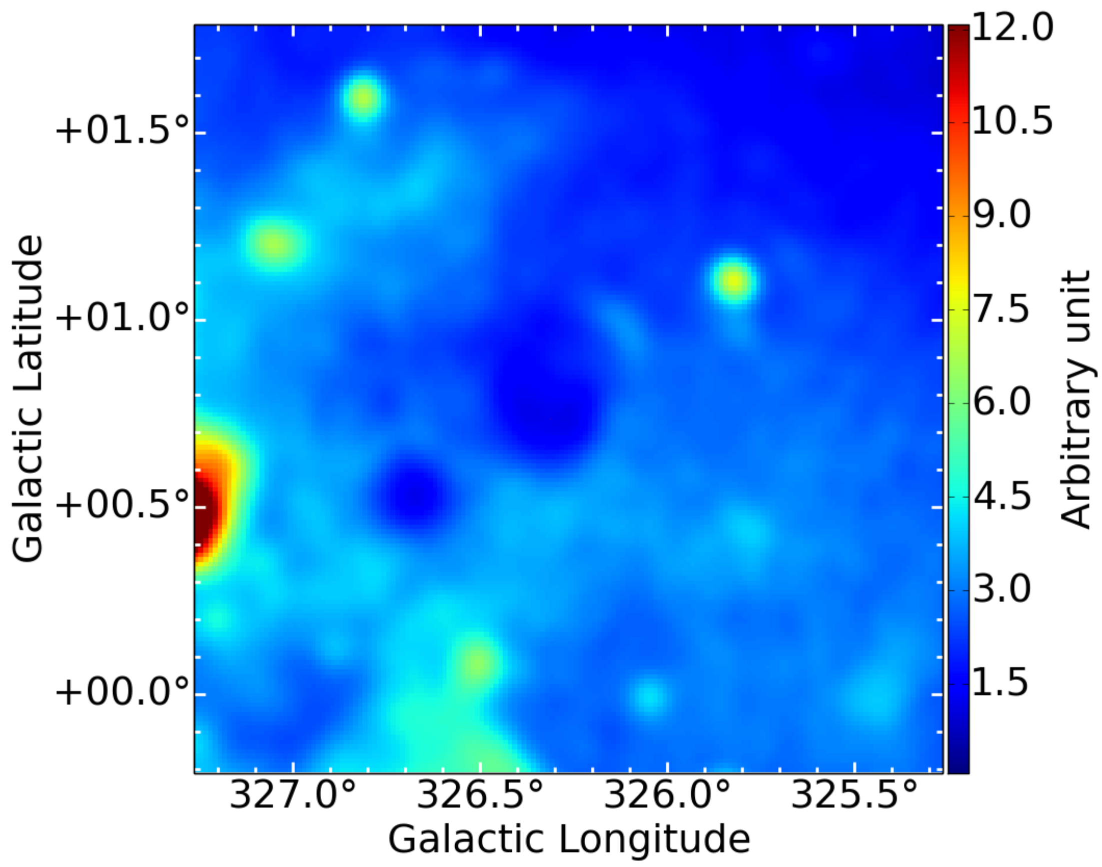

Fig. 3. A small region from the MWA observation on the Galactic Plane.

Fig. 3 is our sample sky region to do extended source finding. Two absorption features are near the center of this image. They are our target sources which cannot find by the existing packages.

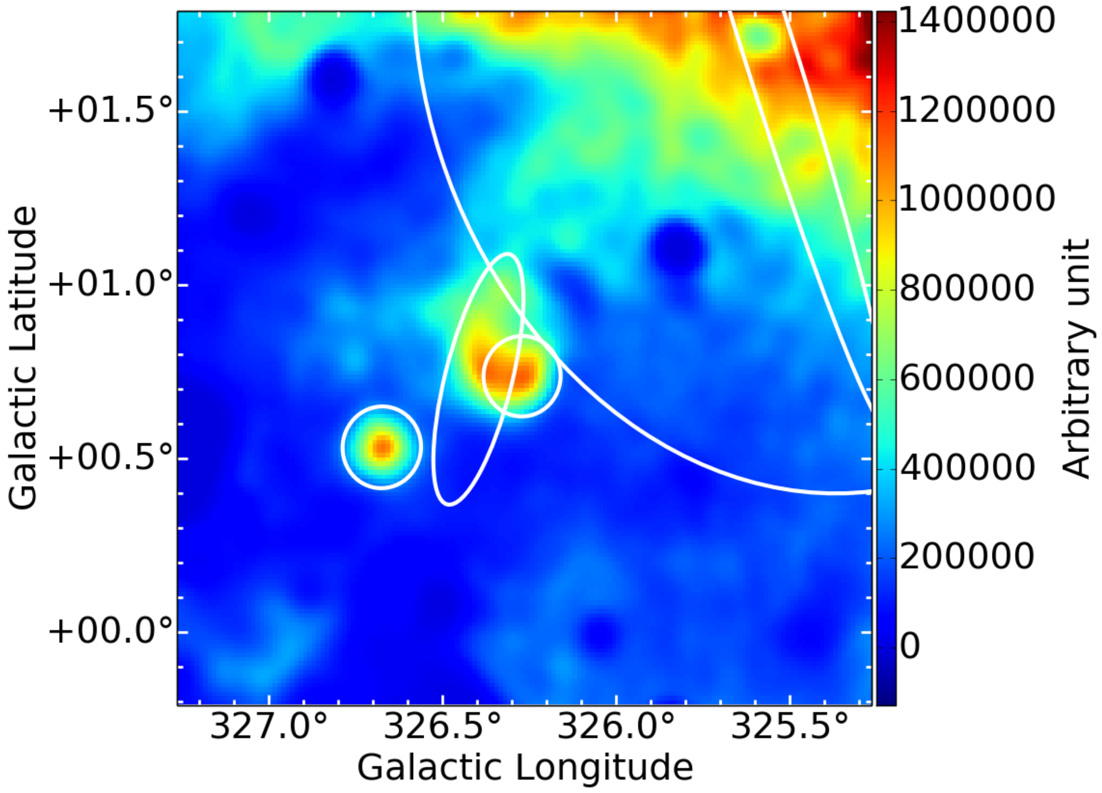

Fig. 4. Initial source finding on the reversed and scaled image.

Fig. 4 is the inversed and scaled image of Fig. 3. The previous absorption features become the emission feature for performing source detection. The white regions show the detected sources using Aegean after our inversion.

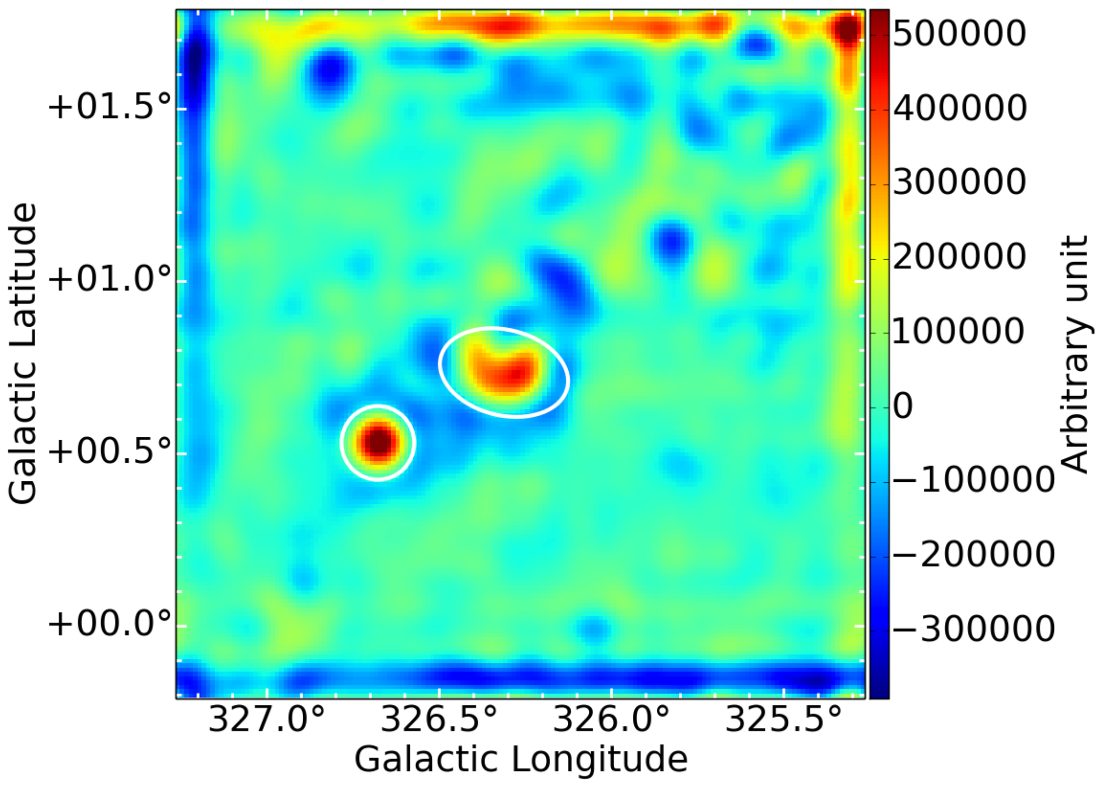

Fig. 5. Source finding results on the image processed by the FFT and inverse FFT.

Fig. 5 shows our source finding results using our image processed by FFT method. The two white regions show the only two sources found by Aegean. This image shows that our FFT processed image improve the accuracy of source finding.

G. Future Work Plan

There are a bunch of work have to be done in following days.

1) Optimize the algorithm to find better data index on GreenPlum and see how far we can go.
2) Try other type of databases, e.g. PostgreSQL Parallel & Distributed version, which will be released at 2017 May & September
3) Try no database solution. Database might not to be the best for big catalog cross-matching, we will try to work on CPU/GPU cluster based on catalog file
4) Complete the next few steps, i.e. cross-matching of new absorption regions with other large survey catalog, for example WISE, GAIA, GLEAM and so on, and from that to create a more precise catalog.
5) current work is based on small scale images, and tests will be extended to larger scale images, with cross-matching also from surveys of various frequencies
6) to improve the source finding algorithm to better detect extended regions and faint sources
7) to integrate parallel processing for the image processing, source detection of large scale images and cross-matching from large database
8) to include more source finding algorithms (e.g. AEGEAN, SOFIA, DUCHAMP) for the performance testing
9) Share our source finding code on the GitHub for helping others

H. Long term goals

After all the works, we hope following target could be achieved:

1) Determine whether using the distributed database or not, at current situation
2) Get a usable big table cross-matching method
3) Generate a comprehensive source detection algorithm and software for the requirements of SKA source detection.
4) Create a comprehensive absorption region catalog using the SKA precursor - MWA.

Thanks for the support of the Chinese/Australian government SKA founding, LOC, SOC, and very appreciated to the observers' advices.