Current Status of the Japanese Virtual Observatory Portal

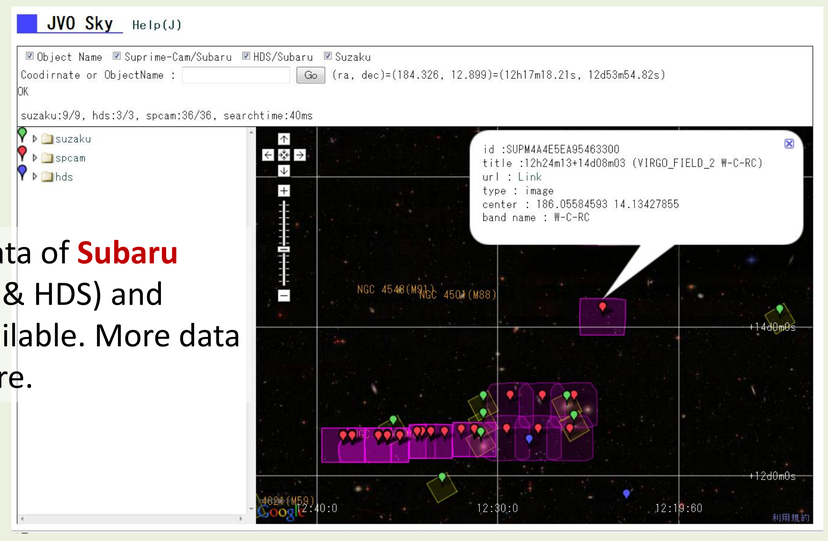
P063

Y. Shirasaki, Y. Komiya, M. Ohishi, Y. Mizumoto (NAOJ),
Y. Ishihara, Y. Yanaka, J. Tsutsumi, T. Hiyama (Fujitsu) ,
H. Nakamoto, M. Sakamoto (SEC)

The Japanese Virtual Observatory (JVO) portal (http://jvo.nao.ac.jp/portal/) is a web portal for accessing astronomical data and analysis system through the Internet. In 2009 and 2010, we developed two new data access interfaces: JVOSky and command-line access interfaces. To enable user to perform all sky search based on SED properties of celestial objects, we experimentally used the Hadoop for performing cross-match of 10 billions of photometric records in the JVO Digital Universe.

JVO Sky

JVOSky is an on-line data discovery service which displays the coverage of observations made by various instruments on the Google sky. Using this interface, a user can graphically find sky regions where data of multi-wavelength observations exist.

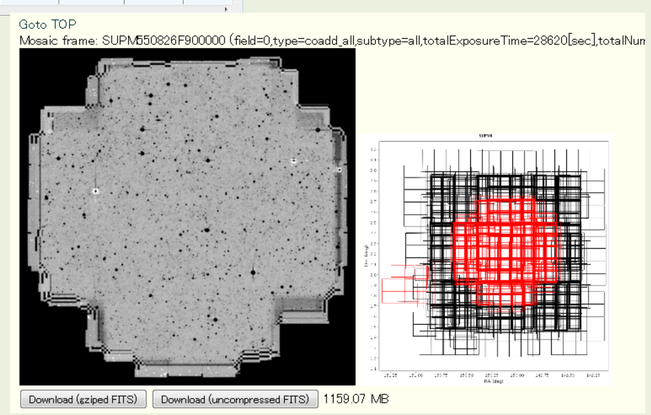


Currently data of **Subaru** (Suprime-Cam & HDS) and **Suzaku** are available. More data coming in future.

Suzaku observations displayed on the JVOSky are linked to the **DARTS** system operated by **JAXA/ISAS**. Suzaku data can be downloaded from there. The other data registered to DARTS (AKARI etc) will be added.

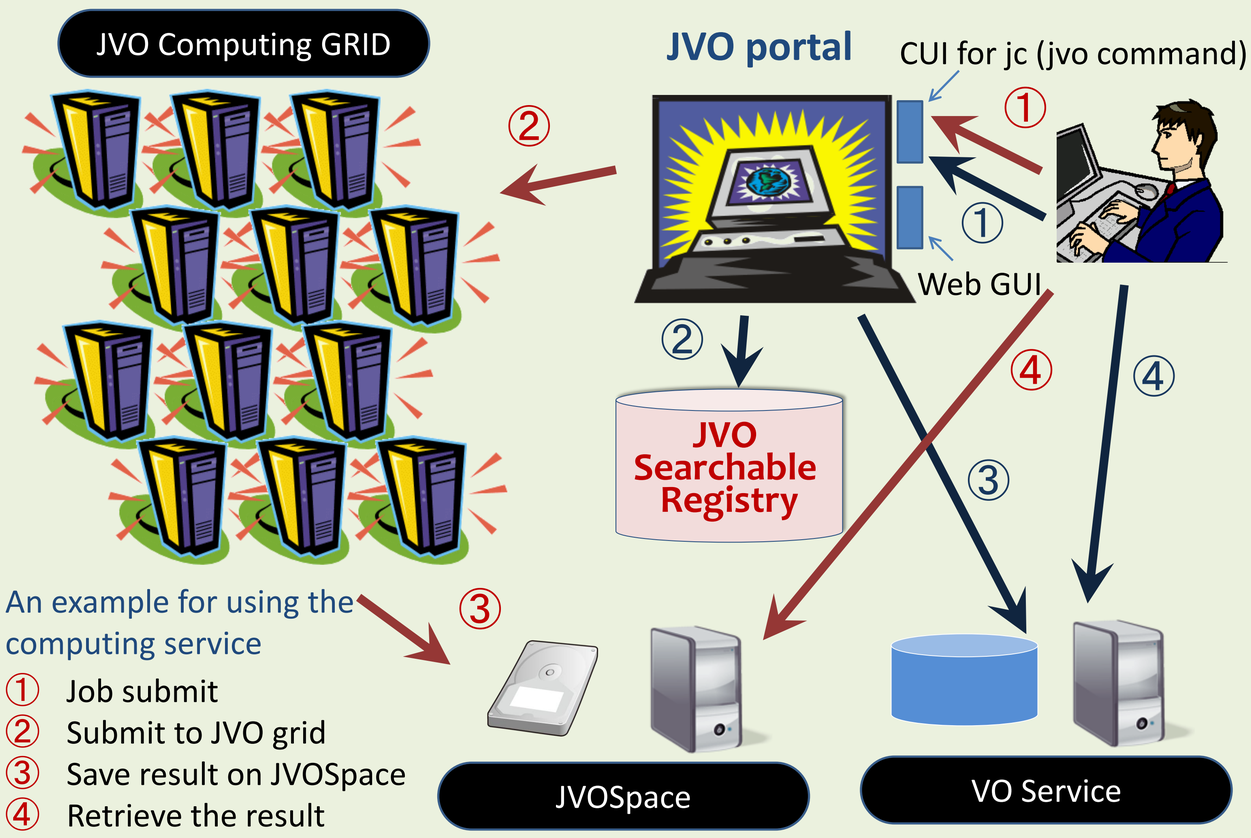
Reduced Subaru Suprime-Cam and HDS are available.

We have a plan to add data crawled from VO services.



Command-line access to JVO

Although a graphical user interface (GUI) is a convenient way for performing a simple query, it is not efficient nor flexible for performing a lot of queries by changing query parameters. Such a situation happens when a user wants to get a large number of data that may exceed the maximum number that a data service can return. We, therefore, have implemented a command line search interface that is accessible through typing commands on the user's computer.



An example for using the computing service

① Job submit
② Submit to JVO grid
③ Save result on JVOSpace
④ Retrieve the result

Syntax of jc (jvo command):

```
jc <command> [<option>] [<argument>]…
```

Examples:

```
jc search –i <jvoql_file>
jc registry –k <keyword>
jc copy2l <source> <destination>
jc run <program_name> <arguments>
jc join –s <votable1> -t <votable2> -o <output> --s-ra
        <RA_column> --s-dec <DEC_column> …
```

Other commands:

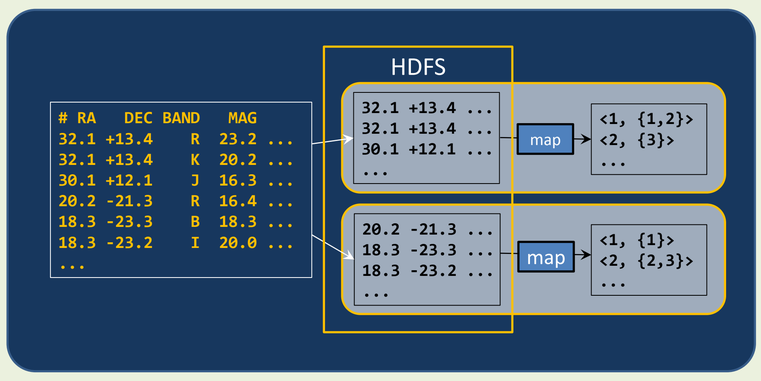
```
ls rsync passwd resume suspent abort ps union join select
```

An example for using the VO query service

① Submit a query
② Search a VO service
③ Search to VO service
④ Retrieve a FITS image

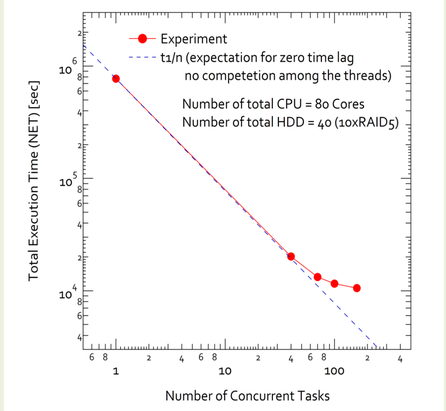
Cross-match using Hadoop

JVO has a huge astronomical database called Digital Universe, which contains coordinates and photometric information of celestial objects collected from major survey catalogs. Currently we provide a functionality to search for data based on coordinates only. However, there would be a science use case where a user wants to search based on SED properties. In order to provide this kind of searching functionality, cross identification among different catalogs should be performed in advance. A search could be conducted against the whole sky, and all the data should be scanned in a reasonable time scale. To achieve such a functionality we are now developing a distributed data search system by means of the Hadoop.



MapReduce for Cross Match

- ✓ Divide the whole dataset into subsets based on a region of sky.
- ✓ The Map function processes whole of the input file to produce cross match result (list of matched record ids)
- ✓ The Reduce function is not executed, since each subset is independent each other.



Experiment

- ✓ 1 billions records (1/20 of whole data)
- ✓ Divided into 6112 files. ~3MB/file
- ✓ Each file contains records of which pos error circle overlaps with the same region specified with an HTM index (level 6).
- ✓ Each file are gziped and copied to HDFS.
- ✓ Max number of task executed in parallel
  1, 40, 70, 100, 160
- ✓ Hardware
  10 servers: each has 2x4 core and 4 SATA HDD

Result

- ✓ If executed by a single task
  9 days for 1G records → 180 days for whole dataset (20G rec.)
- ✓ Parallel execution of 70
  3.7 hours for 1G rec. → 3 days for whole
- ✓ Scaling relation breaks around ~40 tasks
  Overhead of writing to the local FS.
  Writing time occupies ~60% of the total.