Development of the Japanese Virtual Observatory (JVO) prototype

Masahiro Tanaka[a] Yoshihiko Mizumoto[a], Masatoshi Ohishi[a], Yuji Shirasaki[a], Satoshi Honda[a], Naoki Yasuda[b], Yoshifumi Masunaga[c], Yasuhide Ishihara[d], Katsumi Abe[d], Jumpei Tsutsumi[d], Hiroyuki Nakamoto[e], Yuusuke Kobayashi[e], Tokuo Yoshida[e] and Yasuhiro Morita[e]

[a] National Astronomical Observatory of Japan, 2-21-1 Osawa, Mitaka Tokyo, 181-8588 Japan
[b] University of Tokyo, 5-1-5 Kashiwa-no-Ha, Kashiwa Chiba, 277-8582 Japan
[c] Ocanomizu Univerisity, 2-1-1 Otuka Bunkyo-ku, Tokyo, 112-8610 Japan
[d] Fujitsu Ltd., 4-1-1 Kamikodanaka Nakahara-ku, Kawasaki, 211-8588 Japan
[e] Systems Engineering Consultants Co. Ltd., 22-4 Sakuraoka-cho Shibuya-ku, Tokyo, 150-0031 Japan

ABSTRACT

The Japanese Virtual Observatory (JVO) project has been conducted by the National Astronomical Observatory of Japan (NAOJ). JVO aims at providing easy access to federated astronomical databases (especially SUBARU, Nobeyama and ALMA) and data analysis environment using the Grid technology. We defined JVOQL (JVO Query Language) for efficient retrieval of astronomical data from a federated database. We then constructed the first version of the JVO prototype in order to study technical feasibility including functionality of JVOQL, remote operations using Globus toolkit. The prototype consists of several components as follows: JVO portal to accept users' requests described in JVOQL, JVO Controller to parse them into individual query requests, and distributed database servers containing Suprime-Cam data of the Subaru telescope and 2MASS data. We confirmed that this prototype actually worked to access to a federated database. We construct the second version of the JVO prototype system to improve usability, which includes new user interfaces, efficient remote operations, and introduction of analysis tools. In the course of this, Grid service and XML database is employed. In this presentation we describe its design and structure of the new JVO prototype system.

Keywords: Virtual Observatory, Distributed Database, Grid

1. INTRODUCTION

Advance in the technologies of astronomical telescopes and detectors, which are used for recent observational instruments, such as Subaru telescope[1] and Sloan Digital Sky Survey (SDSS), has been raising quality and quantity of astronomical data. Such data archives have become important in the astronomy field. The utilization of astronomical data archives is needed particularly for statistical studies using multi-wavelength data ranging from radio to gamma ray. However, conventional ways of data analysis are inadequate to produce scientific results effectively because of the enormousness and variety of astronomical data. The Virtual Observatory (VO) aims at providing easy accesses to astronomical data archives through the federation of the distributed computer resources over the world

The National Astronomical Observatory of Japan is conducting the Japanese Virtual Observatory (JVO)[2,3] project. We are developing JVO systems to provide astronomers with easy access to federated databases and data analysis systems. JVO is also affiliated with the International Virtual Observatory Alliance (IVOA) and cooperating to establish the IVOA standards. The JVO system is not merely a collection of relational database (RDB) systems, but equipped with capabilities for astronomy. We have defined the JVO Query Language (JVOQL) to describe commands to search for astronomical data in federated databases. And we developed JVO prototype systems which is actually operated by JVOQL commands. This paper describes the outline of the development of the JVO systems.

Further author information: (Send correspondence to M.T.)
M.T.: E-mail: masahiro.tanaka@nao.ac.jp, Telephone: +81 (0)422 34 3561

2. JVO QUERY LANGUAGE

We have defined the JVO Query Language (JVOQL) by extending SQL in order to write query commands to the JVO system. In this section, we describe an outline of JVOQL.

2.1. Naming Rules for Resources

JVO aims at seamless access to databases distributed over multiple organizations. For this purpose, it is necessary to be able to specify required data resources in databases distributed over the world. In the IVOA, a unique naming method for data resources is proposed (IVOA Identifiers). We adopt this rule for identifying tables in distributed databases. However, we employ the dot notation, which is different from the URL or XML notation considered in the IVOA, in order to keep compatibility with the SQL syntax. The notation of a table is as follows.

```
[[authority.]database.]table [[AS] alias]
```

A name given to an organization which manages the database comes at the beginning, and a database (a group of tables) name comes after a dot. This database name is given so as not to overlap within the organization. A table name comes at the end. Under the conditions in which the table name can uniquely identified, names ahead of the table name can be abbreviated.

2.2. Query for Catalog

The astronomical catalog is regarded as a database table and handled with commands written in SQL. However, SQL lacks important features for astronomy, i.e., specification of positions and regions in celestial sphere coordinates, cross match operation, and retrieval of image and spectrum data. Therefore, we considered the following extension to SQL.

2.2.1. Position and Region of the Sky

We consider JVOQL expressions of position and region in celestial coordinate systems. The notation of the position in a celestial coordinate system is defined as follows;

```
POINT( longitude, latitude [, system] )
```

Longitude and latitude is specified with decimal numbers in degrees or the form of "hour:minutes:second, degree:arcmin:arcsec" ("12h34m56.7s, -76d54m32.1s" or "12:34:56.7, -76:54:32.1"). The coordinate system can be specified with the last argument. The Equatorial coordinate system should be accompanied by epoch, e.g., B1950, J2000, and so on.

The REGION phrase is specified using the POSITION phrase as follows;

```
BOX( POINT-phrase, width, height [,rotation-angle] )
CIRCLE( POINT-phrase, radius )
POLYGON( list-of-POINT-phrase )
```

BOX and CIRCLE express regions with the shapes of rectangle and circle, respectively, The coordinate of the center position is provided for the first argument with the position phrase described above. The region size is provided for the next arguments. The size argument can be accompanied by a unit; deg (degree), arcmin (arcminute) or arcsec (arcsececod). If the unit is abbreviated, deg is assumed. In the case of BOX, rotation angle can be specified. The origin of rotation angle is the direction to the North Pole of a coordinate system, and positive means a rotation to the east side. POLYGON expresses the region inside of the border connected with the position list. These region phrases can be described as query conditions in the WHERE clause.

2.2.2. Cross Match (XMATCH)

Cross Match (XMATCH) is an operation to compare two tables and to search for celestial objects which appear in both the tables. This is indispensable to researches using multi-wavelength data. XMATCH is similar to the JOIN operation in SQL. The different point of XMATCH from JOIN is to judge whether they are same object by the closeness of a coordinate position. Since XMATCH is not defined in SQL but indispensable for astronomy, we defined the XMATCH sentence as follows.

```
XMATCH(table1, [!] table2) < accuracy [NEAREST|BRIGHTEST|ALL]
```

"Accuracy" is a maximum distance angle between celestial objects in two tables. In many cases, multiple objects in one table can be XMATCHed with a single object in the other table. The method for narrowing the matched candidates can be specified by one of the following omissible keywords. When NEAREST is specified, the nearest object is chosen among the objects whose positions is within the limits of accuracy. When BRIGHTEST is specified, the brightest object is chosen. When ALL is specified, all the matched objects are returned. The character "!" before the table name means exclusive XMATCH, i.e., the selection of every table1 object which has no counterpart in table2 within the limit specified with "accuracy". This is effective for the selection of objects not visible in visible wavelength but detected in infrared.

2.2.3. Standard Column Name

In general, the column names of a database differ from each other, and one cannot write SQL without knowing the column names. However, astronomical catalogs contains mostly conventional columns, e.g., Right Ascension, Declination, and magnitude. In JVOQL, we decided that the Unified Column Description (UCD) can be specified as column names. UCD is a standard naming rule in the IVOA and is used to classify the column names by their meanings. For example, POS_EQ_RA_MAIN indicates Right Ascension and PHOT_MAG_B indicates B band magnitude.

2.3. Query for Image and Spectrum

Unlike astronomical catalogs, image and spectrum data should be handled in a different way than tables. Image and Spectrum data is expressed as 2 and 1-dimensional grid, respectively. Furthermore, spectroscopic mapping produces 3-dimensional data including both the image and spectrum, which is called as "Data Cube". Support for Data Cube by a common query syntax provides unified query interfaces for astronomy. Hence, we defined a notation to handle image and spectrum data as well as catalog data. Selection of an image with a rectangle region is written in JVOQL as follows;

```
SELECT image.BOX(POSITION-phrase, width, height)
FROM image
```

If the region has another shape, the region phrase described in Section 2.2.1 is specified instead of BOX.(...) form. For spectrum data, a SPECTRUM keyword can be specified instead of BOX.(...) form.

This extension is also used for retrieving counterpart image/spectrum to each selected catalog object as follows.

```
SELECT c.BOX(POINT(c.ra, c.dec), 10 arcsec, 10 arcsec)
FROM catalog c
WHERE BOX(POINT(270,60), 1 deg, 1 deg)
```

In this example, objects in the catalog is retrieved by the region condition in the WHERE clause. Then images of catalog is cut out for every retrieved object.

3. DEVELOPMENT OF PROTOTYPE SYSTEM

One of the purposes of the construction of the JVO prototype is to test the functions of JVOQL described in the previous section. Since it is difficult to implement all the functions of JVOQL, we consider use cases first and construct a prototype system to accomplish the use cases. Then we review the adopted techniques and improve the system.

Currently we have developed JVO prototype systems version 1 and 2. We employed Globus Toolkit version 2 for the prototype 1.[4] However, we found that the elapsed time to execute a basic JVOQL example was more than 10 minutes under the prototype 1. This is because Globus Toolkit 2 tools used for the prototype 1 are not always suitable for a series of right-weight procedure calls. Based on the review of the prototype 1, we developed the JVO prototype system version 2 (hereafter prototype 2). The main purpose of the prototype 2 is to improve performance and usability, by using Grid Service included in Globus Toolkit version 3. In the following sections, we describe the development of the prototype 2.

4. TARGET USE CASE

As an example of JVO use case, we used Subaru Deep Field (SDF) data obtained with Suprime-Cam (2k×4k pixel ×10 CCDs) installed in Subaru Telescope. We used data in two observation wavelength; i' band and z' band. For each band, we prepared images of SDF and catalogs of objects detected from the images. To test the actual federation of distributed databases, we set up database servers, and stored i' band and z' band data into different servers, respectively, and construct a search service, cross match service and image retrieval service.

Target query is "Perform XMATCH between i' and z' band catalog tables, and return a list of XMATCHed objects and counterpart images of the XMATCHed objects" This query is written in JVOQL as follows;

```
select
  i.POS_EQ_RA_MAIN as ra1,
  i.POS_EQ_DEC_MAIN as dec1,
  i.PHOT_SDSS_I,
  z.POS_EQ_RA_MAIN as ra2,
  z.POS_EQ_DEC_MAIN as dec2,
  z.PHOT_SDSS_Z,
  i.BOX(POINT(ra1,dec1), 0.005, 0.005),
  z.BOX(POINT(ra2,dec2), 0.005, 0.005)
from
  naoj.sdf.i i,
  naoj.sdf.z z
where
  XMATCH(i, z) < 10 arcsec NEAREST and
  BOX(POINT(201., 27.4), 0.10, 0.10) and
  i.PHOT_SDSS_I - z.PHOT_SDSS_Z > 0.0 and
  i.PHOT_SDSS_I < 20.0 and
  z.PHOT_SDSS_Z < 20.0
```

In the WHERE clause, the XMATCH condition means cross match between tables i and z. The BOX(...) condition indicates search for celestial objects within a 0.1 degree square region centering on the point at (RA, Dec) = (201 degrees, 27.4 degrees) in the Equatorial coordinate system. In the SELECT clause, the columns of i.BOX(...) and z.BOX(...) indicates retrieval of i' band and z' band images, respectively, with 0.005 degree square regions centering on the position of celestial objects matched in the WHERE clause.

Once the JVO system receives this query, it parses the JVOQL sentence, generates a work flow, calls remote services in distributed servers, receives result catalog and images, and stores them to a user directory in the portal server. The result data can be displayed by the WWW browser in various ways, or it can use as input data of an analysis server further.

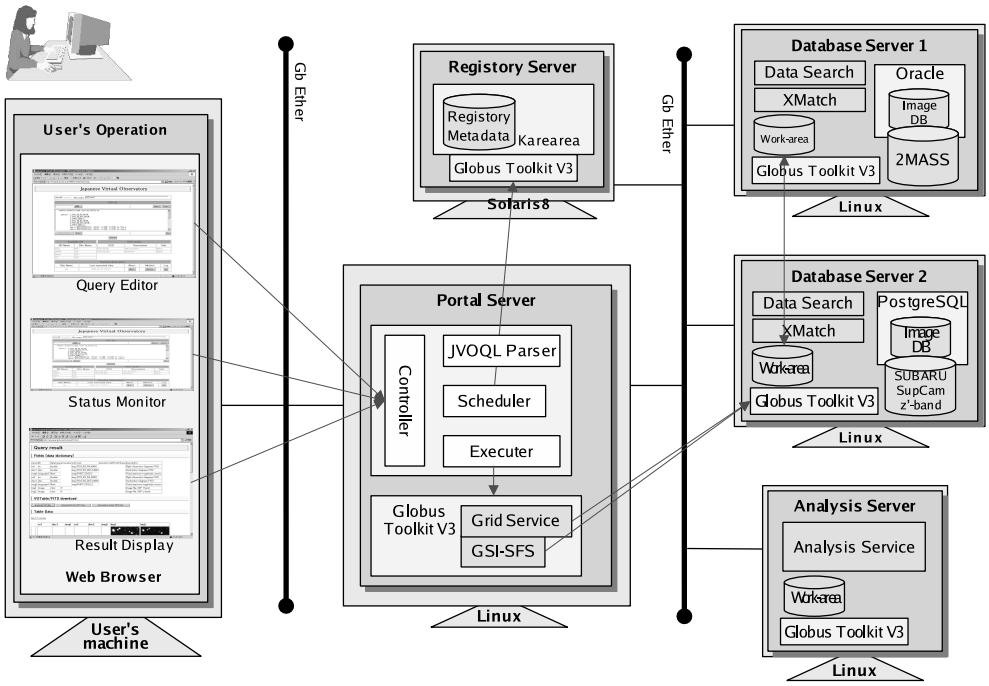

Figure 1. Schematic view of the configuration of JVO prototype system.

5. SYSTEM DESIGN

5.1. Overview

The system configuration of the JVO prototype is shown in Figure 1. At first, a user accesses to the JVO portal server using a WWW browser, and sends query commands written in JVOQL to the execution controller (Section 5.2). The controller analyzes the commands, finds servers which offer a requested service by consulting the registry (Section 5.4), and creates a "work flow". Based on this work flow, remote procedures are called with toolkits of Grid (Section 5.3).

The query in Section 4 is decomposed into the following work flow.

1. Send each query to two servers and acquire the number of celestial bodies which matches conditions.
2. Perform query to the server where few celestial bodies are matched. The result is sent to the next server.
3. On the next server, perform a cross match operation between the result table of 2. and the table in this server.
4. Perform image query based on the result of the cross match.
5. All the result data are moved to the portal server.

The reason that count queries are performed before real query is to ensure less amount of data transfer.

5.2. Execution Controller

The operation of the JVO, which is overviewed in Section 5.1, is controlled by the JVO controller in cooperation with the following subcomponents; **JVOQL Parser**, **Scheduler** and **Executer**. The controller and these subcomponents are written in Java language, and located in the portal server. The JVOQL parser analyzes the JVOQL syntax and decompose into query elements. The Java code to parse the syntax of the JVOQL is

generated with JavaCC (Java Compiler Compiler). The scheduler takes the output of the parser and generate a work flow which includes job elements for individual servers. Based on the work flow, the controller calls the executer which perform remote procedure calls.

5.3. Remote Execution/Data Transfer

The federation of astronomical databases distributed all over the world requires a standard of remote procedure calls. For this purpose, we considered the application of Globus Toolkit developed by Globus Alliance and verified its applicability to the JVO.

As described in Section 3, remote procedure calls of the prototype 1, for which Globus Toolkit 2 is used, took too long time. To improve performance, we employed "Grid Service" introduced in Globus Toolkit 3 for remote procedure calls of the JVO prototype 2. The Grid Service is based on the Web Service. Its communication protocol is SOAP, which is not necessarily suited for transferring large binary data. Hence, we employed RFT (Reliable File Transfer) in Globus Toolkit 3 for data transfer.

We employed standard format for data transfered between distributed servers in order to achieve future connection with other VOs easily; VOTable for astronomical catalog data, and FITS (Flexible Image Transfer System) for Astronomical image and spectrum data.

5.4. Registry

In order to retrieve data from distributed data servers, the controller needs to know in which server the requested data are stored. Information on distributed servers and astronomical data stored in them is a kind of metadata. In JVO, we refer a registry to a facility which collects metadata and provides a service to search metadata.

We employed the UDDI (Universal Description, Discovery, and Integration) as a registry for the prototype 1. UDDI is a tool for finding services in the Web service framework. However, UDDI does not necessarily fit search services for complicated astronomical metadata. It does not provide any capability to search by observation wavelengths or regions of the sky. For the prototype 2, we redesigned the metadata of astronomical catalogs and stored them into a registry. The data structure of metadata is designed so as to keep compatibility with the metadata standard proposed in IVOA. We stored these metadata into an XML database product "KAREAREA", and enabled query with XPath. We plan to use the OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) for exchanging metadata between data servers and a registry server.

5.5. Data Server

The data services, i.e., search and cross match of astronomical catalog and retrieval of astronomical image and spectrum data, have been deployed on each data server. RDBs are used for construction of an astronomical catalog search service. One of the purposes of this prototype development is to test the federation of heterogeneous distribution databases. Hence, two kinds of RDB, Oracle and PostgreSQL, are used as database management systems. Heterogeneous operating systems, Solaris an Linux, are used as the test environment. We developed functions which lack in RDB, including the interpretation of region syntax and UCD and the output in the VOTable form. The generation of VOTable requires column metadata composed of column name, data type, unit, UCD, and so on. In addition, in order to search by celestial spherical coordinates efficiently, we employed HTM (Hierarchical Triangular Mesh) which expresses triangle regions of the celestial sphere coordinate with a 1-dimensional index. A cross match operation is performed between two tables; one is VOTable, the other is a table in RDB. We developed program components for cross match operation and VOTable reader, and applied them to the simple search service. We also developed services for extracting FITS image and spectrum data of the celestial object listed in VOTable.

6. USER INTERFACE

The structure of the JVO system is complicated. However, a JVO user, who is supposed to be an astronomer, is not always fluent in computers. In order for every astronomer to be able to manipulate the JVO system without the knowledge about the JVO system, we implemented instructive user interfaces to the JVO system

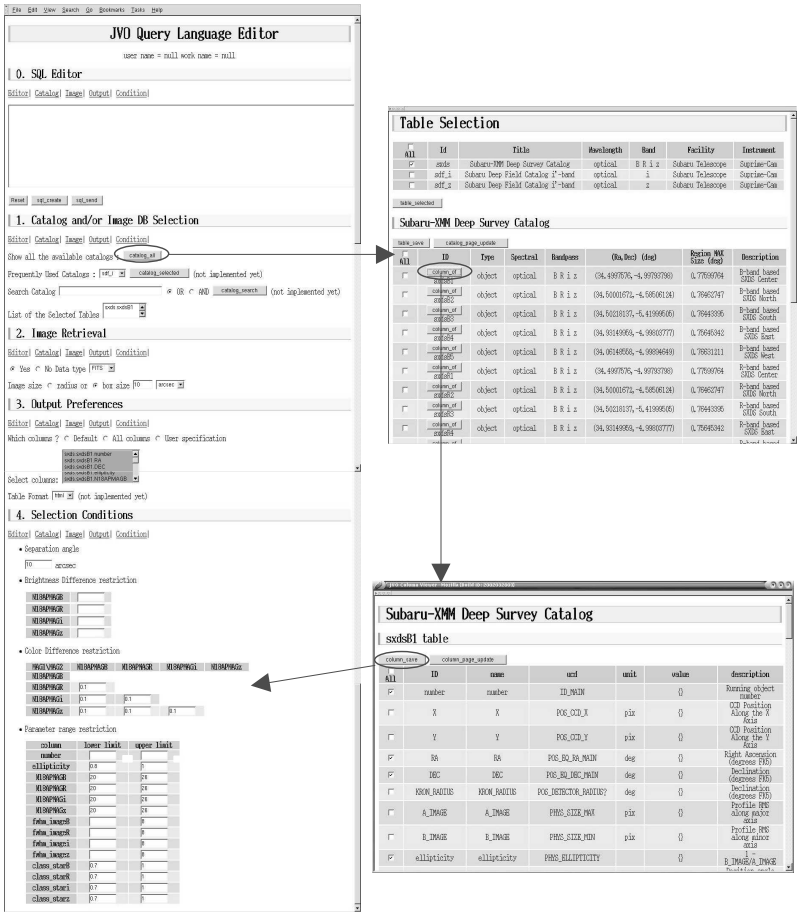

Figure 2. JVOQL Editor windows. Users can browse and choose available tables and columns from the list of the right windows. The result is reflected in the parameter filling window (left). By filling parameters, a JVOQL sentence is generated automatically.

on the WWW. We employed "Struts" for the the framework of Web application and JSP (Java Server Pages) for writing interface pages.

The user interface of JVO consists of **JVOQL Editor** and **Data Viewer**.

One feature of the JVOQL editor is the functionality of search for available data in the JVO system. The user interface provides easy access to lists of catalogs, images, spectra searched by various keys, e.g., radio, X-rays, and so on. This is enabled by search for metadata into the registry (Section 5.4). Another feature is the assistance for JVOQL writing (Figure 2), in order to write JVOQL without the knowledge of JVOQL. By displaying the list of the columns contained in a table, choosing it, and filling in parameters, a JVOQL sentence is automatically generated.

The features of Data Viewer (Figure 3) include VOTable viewer, a quick-look image/spectrum viewer, a color-color plot, and a three-color synthesis. The result data can be browsed in various ways, downloaded to user's machine, and used as input data for analysis services. As an example of a analysis service, we implemented a service to search for gravitational lens objects.[5]

7. ASSESSMENT OF PROTOTYPE

We tested two queries which took 2.3 and 13 seconds as actual processing times in remote servers, respectively. Their elapsed times including remote procedure calls was 2.8 and 16 seconds, respectively. The measured

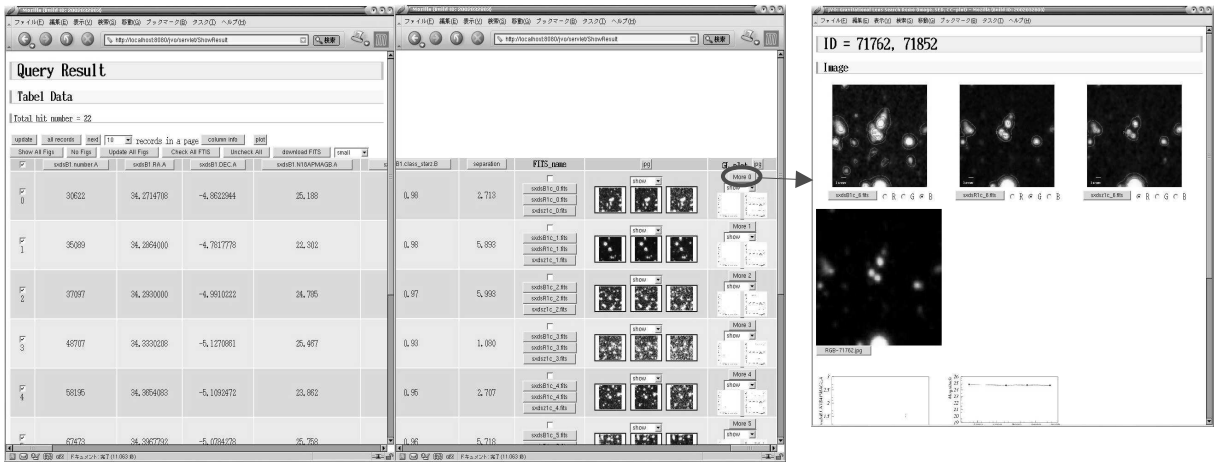

Figure 3. VOTable viewer windows. The left two windows show a VOTable and quick-look images. The right window shows a synthesized image made of three images of different bands.

overhead time was only around 30 ms. Thus the elapsed time of the prototype 2 becomes shorter than that of the prototype 1. This is because the latency time in the prototype 1 due to a polling interval is eliminated. This result indicates that the prototype 2 is improved in performance using the Grid service and can be a basis of the practical system.

In the implementation of the JVO prototype 2, we found that the Grid service was not equipped with any function to transfer large binary files efficiently. Hence we employed RFT for data transfer. However, it is a cumbersome work for a service provider, who can be an astronomer, to prepare a service using both Grid service and RFT. It will be helpful if middleware provides a protocol with capabilities of both remote procedure calls and efficient file transfer.

ACKNOWLEDGMENTS

This research was supported by Grant-in-aid "Information Science" (15017289) carried out by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

REFERENCES

1. N. Kaifu, "Subaru telescope," *Proceedings of SPIE* **3352**, pp. 14–22, 1998.
2. M. Ohishi, "Construction of the Japanese Virtual Observatory (JVO)," *The Astronomical Herald of Japan* **95**, pp. 566–575, 2002.
3. Y. Mizumoto, M. Ohishi, N. Yasuda, Y. Shirasaki, M. Tanaka, Y. Masunaga, K. Miura, H. Monzen, K. Kawarai, Y. Ishihara, Y. Yamaguchi, and H. Yanaka, "Construction of the Japanese Virtual Observatory (JVO)," in *Astronomical Data Analysis Software and Systems XII*, H. E. Payne, R. I. Jedrzejewski, and R. N. Hook, eds., *ASP Conf.* **295**, p. 96, 2003.
4. M. Ohishi, Y. Mizumoto, N. Yasuda, Y. Shirasaki, M. Tanaka, S. Honda, Y. Masunaga, K. Miura, H. Monzen, K. Kawarai, Y. Ishihara, Y. Yamaguchi, and H. Yanaka, "A prototype toward Japanese Virtual Observatory (JVO)," in *Astronomical Data Analysis Software and Systems XIII*, 2003.
5. Y. Shirasaki, Y. Mizumoto, M. Ohishi, N. Yasuda, M. Tanaka, S. Honda, H. Yahagi, M. Nagashima, G. Kosugi, N. Kashikawa, E. Matsuzaki, F. Kakimoto, and S. Ogio, "Searching for a cosmic string through th gravitational lens effect: Japanese Virtual Observatory science use case," in *Astronomical Data Analysis Software and Systems XIII*, 2003.