Japanese Virtual Observatory (JVO) Prototype 2

Masahiro Tanaka, Yuji Shirasaki, Satoshi Honda, Yoshihiko Mizumoto, Masatoshi Ohishi (NAOJ), Naoki Yasuda (U. Tokyo), Yoshifumi Masunaga, (Ochanomizu U.), Yasuhide Ishihara, Katsumi Abe, Jumpei Tsutsumi (Fujitsu Ltd.), Hiroyuki Nakamoto, Yuusuke Kobayashi, Tokuo Yoshida, Yasuhiro Morita (SEC Ltd.)

URL: http://jvo.nao.ac.jp/

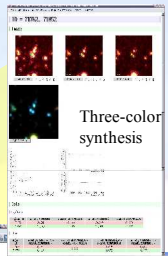
ABSTRACT
We describe the architecture of the Japanese Virtual Observatory (JVO) prototype system version 2. JVO aims at seamless access to astronomical data archives stored in distributed data servers as well as data analysis environment. For this purpose, it is important to build a framework for access to remote servers, including remote procedure calls (RPCs) and data transfer. A data request for distributed database is written in the JVO Query Language. The JVO system parses the query language, decomposes it into individual remote procedures such as retrieval of catalog, image and spectrum and cross matching, and generate a work flow. Based on this work flow, remote procedures are called. For RPCs of JVO prototype system 1, we employed Globus toolkit 2 (GTK2). However, latency time of GTK2 RPCs was too long for successive short-time jobs. Therefore, we employ Globus toolkit 3 (GTK3) for JVO prototype system 2. As the result, we find that Grid Service in GTK3 improves performance of RPC. In addition to Grid Service, Reliable File Transfer (RFT) is used for efficient data transfer. Astronomical data stored in distributed servers are discovered through a registry server which provides metadata discussed in the IVOA registry working group and is built using a XML database.

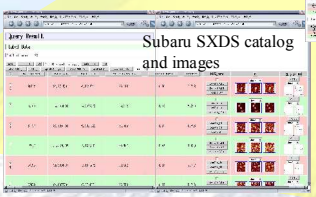
Configuration of JVO Prototype 2

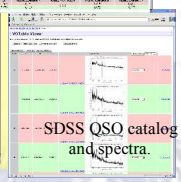
The main purpose of the JVO prototype is to test functionality of JVOQL and employed technologies. This description is based on JVO prototype version 2.

Query Interface
You can choose catalogs and specify query conditions on WWW browser. Then JVOQL sentence is automatically generated.

Result Display
You can browse search results (VOTable, FITS) on your WWW browser with various tools (table viewer, three-color synthesis, color-color plot, etc).



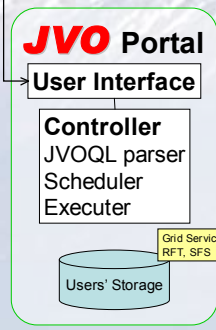




Controller
Components
- "JVOQL Parser" generates query for each host
- "Scheduler" generates schedule based on execution results
- "Executer" executes services on remote hosts

Description of Workflow
- Count queries can be executed in parallel.
- Search and XMatch service can be called sequentially.

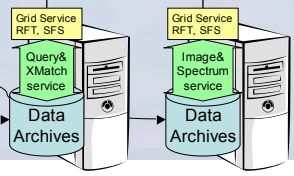
→ "Dependency" is considered.



Registry
is a facility to maintain and provide **metadata**, i.e., information on remote servers, services and observational data archives. Metadata is expressed in XML form, and stored into XML DB.

Grid
- Distributed servers of JVO are federated using Grid.
- Globus Toolkit ver.3 is employed as a Grid middleware.
- Grid Service is used for remote procedure calls.
- RFT (Reliable File Transfer) is used for data transfer between data servers, and SFS (Self-certifying File System) for transfer between server and portal.

Subaru
ALMA
SDSS
2MASS
...



Data Server
Subaru, SDSS and 2MASS data are stored into distributed data servers using RDB. Query services are implemented to handle JVOQL-specific functions and output VOTable. XMatch service and Image/Spectrum retrieval services are also implemented. These services are called by Grid Service.

Analysis Server
As an example of research using JVO, We implemented a service to search for gravitational lens objects from Subaru data.

Development of JVO Prototype 2

JVO development
JVO Project start — April 2002
Prototype 1 finish — March 2003
Prototype 2 (this poster) finish — March 2004
Prototype 3 — under development

Data Discovery
- Proto1: UDDI
  - –UDDI is for "Service discovery", not for "Data discovery"
- Proto2: XMLDB
  - –XMDB product "Karearea" is used.
  - –XPath search
  - –enables both "Data discovery" and "Service discovery"
  - –Metadata contents are based on IVOA standard

Performance of Grid Execution
Proto1:
- –Globus toolkit ver 2
- `-globus-job-run` command is used
- –1 call = 30 sec — **slow!**
  - •1 query ~10 min!

Proto2:
- –Globus toolkit ver 3
- –using Grid Service
- –1 call = 2-3 sec — **fast!**
  - •overhead time is only ~ 30 ms

Data Transfer
- We tried:
  - –RFT (Reliable File Transfer; GridFTP)
  - –GSI-SFS
- We employed RFT
  - –SFS is not flexible (A server cannot be a client).
- However,
  - –Need support for HTTP and Web Services to interoperate with IVOA.

Plans toward Prototype 3
- support SIAP, SSAP, SkyNode
- implement ADQL
- employing OAI-PMH architecture
- flexible workflow architecture
- introduce User management
- etc.