PDS Search Services

Distributed Infrastructure Design Team

June 3, 2009

Overview

This document provides a description of several representative search systems within the Planetary Data System (PDS). It is intended to provide background information to assist in the design of the PDS 2010 search service architecture.

Engineering Node Search

The PDS has a central high-level catalog or "data search" database that is used to select data sets based on a set of global metadata parameters (e.g. target, mission, instrument, data type). There are several JSP forms-based interfaces to what is called the "catalog profile server" [which accesses the Sybase catalog database]. This catalog is built by the engineering node staff from metadata supplied by the discipline nodes. A full-text search capability is also provided using Solr and Lucene to index the catalog database and a separate file that describes the search tools at the discipline nodes. The search systems provide a list of data sets with links to data set search systems at the discipline nodes or to locations where the data set volumes can be perused.

Planetary Image Atlas II

The Planetary Image Atlas II (PIA) is a search and retrieval tool for planetary images. The Atlas web service is a demon server that maintains in memory all mission configurations and is accessed to provide mission specific or multi-mission views of the configuration. The Atlas client provides the view request when first accessed by the user. A subset of the configuration is then used by the server to perform transactional queries from the client. The configuration file is used to build all the search menu screens and to provide all relevant database locations and protocol specifications as well as schema mapping, searchable parameters and returnable parameters. The user interface provides tabs for Intro[duction] to Atlas, QuickSearch, Product Search, Geometry Search, Instrument Search, Feature Search, Time Search, Map Search (with text search) and a search results page. Atlas II is deployed on multiple servers running Red Hat Linux. It is a three-tiered architecture with a presentation layer, core layer and back-end layer. The presentation layer uses the Google Web Toolkit (GWT), AJAX and RPC as a bridge between the GWT AJAX client to the Glassfish SOAP JAXWS 2.0 server.

The user interface provides tabbed input screens for search categories that vary by mission and instrument. It also provides a text based search suggest box that pops up suggested

keywords and values when text is entered. This will be expanded to integrate the USGS gazetteer for feature name searches. Google Maps is used to provide a map-based geographic selection screen. The core layer includes three web services: atlasservice, which performs searches based on user input to the search screens; atlascartservice for ordering and downloading data; and atlasingestservice for updating Atlas metadata. The atlassearch service translates user input into either OODT DIS or SQL query and sends it off for execution using either JDBC, OODT webgrid servers or Tomcat DBCP for accessing the MySQL and PostGres servers with connection pooling. It then combines the results and presents them to the user. Atlas allows for downloading of products using an OODT product server, wget scripting, or FEI for bulk downloads. The backend layer includes metadata and data storage components. These include a MySQL database for the image node inventory; PostgreSQL database for Unified Planetary Coordinate (UPC) database access; Distributed OODT profile servers and distributed OODT product servers. A big issue is trying to understand the semantically different interpretations of keywords from mission to mission and trying to unify the queries across missions is very difficult.

Orbital Data Explorer (ODE).

The ODE system was initially designed to provide integrated access to multiple Mars Reconnaissance Orbiter instrument product collections. This capability has been extended to add data from other Mars missions and instruments. Subsequently a separate Mercury ODE has been released to support Messenger data access and a Lunar ODE is planned. The major ODE system components include the web interface, ODE metadata database, the backend processor (Bunter). Other components include the Geosciences Common Subsystem library and the data archives, which are not part of ODE. The system software is developed with commercial software tools including Microsoft Visual Studio 2008 (IDE and C#), SQL Server Management Studio and Visual SourceSafe software management system. The web interface uses ASP.NET, ASP.NET AJAX, HTML, Javascript, ESRI ArcGIS Web ADF, MS SQL 2005, and internally developed Geosciences DLL files. The web site is hosted on MS Internet Information Services 6.0 with .Net framework 2.0, ASP.NET 2.0 AJAX Extensions 1.0 and ArcGIS Server Web ADF Runtime for Microsoft .Net framework. Code is written in ASP (Active Server Pages) and C#.

The main components of the ODE web interface include the Home page, Data Product Search, Tools links, Data Set volume browser, Download system and Help system including user forum. Product Search requires selection of data sets to search, followed by additional filtering, on product id, product center location and time range. Results can be provided in a tabular listing with browse renditions or their locations plotted on a map display. Results can be selected for inclusion in the user "shopping cart" on the tabular listing. The user can click on selected products and have browse versions displayed in a Zoomifyer frame, which can also display metadata, labels or related products. Having the results plotted on the map display allows the user to graphically select a subset of products before returning to the tabular display. The Tools menu includes a link to a special data product search for coordinated observations, a link to a search page for a Mola PEDR query and a link to information about displaying observation footprints with Google Earth/Mars. The Data Set

Browser tab provides a volume browser, which can access a local file system, or remote data either on an http server, ftp server or PDS-D server (this is being phased out). The Download option allows products to be viewed in the browser or to be packaged for bulk downloading at a later time. The user can select to have ancillary files included in the package.

The ODE metadata database holds metadata for an ODE instance. The database components include an sql relational database, extended labels, browse data, thumbnails, zoomifyer files, map files, geographical names and configuration files. The database is accessed by sql queries or C# access classes. The database is built using the backend processor, Bunter (written in C#) which uses a set of configuration files which provide the location and mechanisms for accessing its sources of data and metadata. The metadata database is built by scanning the product labels from the source data sets. A major issue is adapting the system to the many variations of PDS products developed by different data producers ("Processing products from different datasets, different eras, different nodes, different producers can be very challenging".)

Rings Multi-mission Search

"The Rings Node Multi-mission search system (description provided by Mark Showalter) is hosted on a Mac Xserve and implemented with MySQL, AJAX, PHP. The search consists of SQL tables with a common primary key corresponding to an "observation". An observation might encompass several related products (such as raw and calibrated versions of the same image). Every table contains a particular set of parameters that you might want to constrain, corresponding to one (or sometimes two) tabs on the query form. For example, one table contains general parameters, one contains Cassini-specific parameters, one contains image-specific parameters, and one contains Cassini ISS-specific parameters. The general table contains a record for every observation in our database; the Cassini table contains a record for every observation from Cassini (whether ISS, CIRS, VIMS, etc.); the image table contains a record for every image in our database (whether Cassini, Voyager, Hubble, etc.). Each time the search is narrowed down, the engine performs a query on the general table, and then (for example) determines which missions are represented in the list returned. If the list returned only contains observations from one mission, e.g., Cassini, then the Cassini tab appears. If only images are represented, then the image tab appears. A separate trigger table lists the conditions that must be satisfied for each particular tab to appear. The net result of all of this is a kind of multiple inheritance. Adding a new data set entails populating each of the existing tables with the relevant information. Any parameters left over, which are specific to that particular data set, go into their own new table. Then the trigger table is updated to describe when those parameters should appear, which is basically when a search has been narrowed down to observations that only reside in that data set. Once the user requests results, the system does a query on a separate table of data products to determine what files are associated with each "observation". This makes it possible for an observation to comprise multiple products, and also for a product to contain multiple observations. Note that the forms themselves are also generated using queries on the tables. For example, if we add a new mission "New

Horizons" and a new instrument "LORRI" to the general table, then these options will immediately appear the next time any user loads the general form. Data ingestion is done using Perl scripts, which are customized for each dataset to be ingested.