General Idea

The EasyThredds service operates as a proxy server on the THREDDS data server to facilitate data access across protocols and queries. This implementation defines its own query language and is capable of translating into CdmRemote[1], OPeNDAP[2] and NCSS[3]. Unfortunately, none of the testing service worked properly with the DAP4 protocol. In theory, a similar approach as with OPeNDAP should be applicable. Check the presentation for a visual representation of the components.

Packages

The project is structured using the following packages:

- **Config:** Contains utilities for accessing the properties
- **Protocol:** Contains specifics for reading, translating and sending data using different protocols
    - **Parse:** Provides details on how to parse different inputs (ranges)
    - **Reader:** Implementations for reading data using each of the supported protocols
        - **NcssMeta:** Specific classes to help interpret the XML meta data returned by NCSS
    - **Translated:** Translations from the custom query language to each of the protocols
        - **Decision** and **Decision.Nodes**: The decision tree for selection of the best protocol
        - **Util:** Utilities such as data holders and data interpretation
- **Resource:** The RESTful service interface
- **Service:** Controller classes providing the high-level functionality of EasyThredds

All classes are annotated with JavaDoc for further details. Additionally the test directory contains both simple unit tests and higher-lever programs:

1. <u>PerformanceComparison</u>: takes multiple queries to be executed repeatedly and plots the time measurements using a boxplot
2. <u>RandomPerformanceComparison</u>: generates multiple random queries, executes each query for each protocol, measures the execution time and finally returns which protocol performed best

Technologies

I used Java 8 for the implementation and the Jersey framework to provide the RESTful service (cf. the web.xml).

Maven dependencies:

- Jersey - JAX-RS Reference Implementation for the RESTful webservice
- JodaTime for date and time handling
- JFreeChart to create the boxplot
- edu.ucar.* for THREDDS and NetCdf tools
- commons.io for easy file downloads

[1] http://www.unidata.ucar.edu/software/thredds/v4.5/netcdf-java/reference/stream/CdmRemote.html
[2] http://www.unidata.ucar.edu/software/thredds/v4.6/tds/tutorial/DAP.html
[3] http://www.unidata.ucar.edu/software/thredds/current/tds/reference/NetcdfSubsetServiceReference.html

- commons.math for simple statistics
- etaprinter for the progress bar during the random query evaluation

Further Tasks

- As mentioned above, the DAP4 protocol & reader could not be tested as none of the used catalogues provided a working service.
- In order to read the stream data returned by CdmRemote (see CdmRemoteReader) generics had to be used to make the data accessible. Unfortunately, the provided implementation of CdmRemote (from the according edu.ucar.* package) was not compliant with the process of the server. For instance, when requesting a subset variable the service returns a data stream but the client expects a header stream. Thus, the wrong MAGIC_START sequence[4] is returned.
- Currently, the first time a query requesting variables x,y,z is sent, the meta data (or coordinate data) is retrieved for these specific variables on the provided data set. The next time the same data set is queried with of the known variables, the cached information is used. However, if a new variable is requested on this data set, the procedure will fail as it does not yet provide the information. A better solution would be to request all coordinate data on the data set and then use only the necessary parts.
- There is no cache invalidation routine so far. Once the service starts / restarts, the cache is emptied and new coordinate data is requested.
- Currently, the user sends a request, the proxy performs changes and redirects the final result to the user without looking at it in detail. An improvement could be to download the data in parallel and to perform further requests (e.g. subsetting) on the downloaded data set.
- There are better metrics to determine the performance of a currently processed query. (measure: translation, download and full iteration instead of the translation only)

[4] http://www.unidata.ucar.edu/software/thredds/v5.0/netcdf-java/reference/stream/NcStreamGrammer.html