A data storage strategy for generic, heterogenous scientific data

Systems for storing and managing scientific data are often targeted at specific data types, and often have specific applications in mind. This focus makes it possible to build integrated systems that are tailored to important problems and with implementations that cater to the experts in the field.

The downside is that systems are often expensive and difficult to extend. In quickly developing fields (like molecular biology) there are large volumes of data and a plethora of formats and technologies, and future uses can be hard to predict. As these data are scientifically important, there is a need for lightweight systems that can incorporate these data with a minimum of effort, yet maintain enough structure to make the data reusable and interpretable in the future.

Goals

We have therefore designed a system with the following goals in mind:

The system should be *generic*, or data agnostic. It should not be limited to specific data types, and adding new data types and formats should be as effortless as possible.

The system should be *simple* to understand. It should be easy for data managers to handle data without explicit understanding of the data, and it should be equally easy for domain experts to use data without much knowledge about data management structures and systems. This implies a *separation* of data management skills from domain knowledge.

It is important that the system is *usable*. This means that scientists can easily obtain copies of relevant data, in formats they are used to, and with sufficient metadata to ensure *integrity* and *provenance*. Similarly, it must be a minimum of effort for the scientist and data manager to prepare new data sets for inclusion.

The system should be *long term* sustainable. This means it should be as *technology independent* as possible, and not tied to any particular language or software. It should be *modular* and *forwards compatible*, so that existing functionality will not be affected by new developments in data or metadata formats.

Structure and implementation

Each data set is a stored in a unique directory, and the data is stored in files using native file formats. This makes it simple to extract or transfer datasets (with a file copying operation, or by exposing the directory to an HTTP server and using e.g. `curl` or `wget`). Receiving data in standard formats also relieves the scientist from learning about any structure imposed by the storage system.

Metadata is stored in a separate file (simply named `meta.xml`) using a simple XML schema. The metadata consist of a list of files contained in the data set, including checksums and file types. In addition, there is a free text description, which allows markup of specific terms (e.g. names of species, dates, locations, or references to other data sets). Using XML markup lets the system enforce standard lexical formatting and reference notations (e.g. using TSN numbers to uniquely identify species). An important part of the metadata is *provenance* of the data, how did the data set come about.

Many central functions (including: search, visualization, automated analysis, data conversion, data extraction, user management) are intentionally omitted from the core specification of the system. By implementing these as separate, external modules, these can be tailored to specific needs or domains, and ignore data and metadata that are irrelevant for their purpose.

Use cases

Verification and validation of data sets include checking metadata contents and data formats. An XML validator is used to ensure that the metadata file is well-formed and that the lexical format of formal values is correct. Additional checks, like md5 checksums, are performed by ad-hoc scripts.

Downloading a copy is performed with standard software operating on files, for remote access using HTTP (`wget` or `curl`), SSH (`scp`, `rsync`).

Searching metadata is performed by scanning metadata files and building a database (using `xapian`). A CGI front end provides a web interface.

Data submission is currently manual, and requires some intervention by a data manager in order to construct an appropriate metadata file. This can also be achieved through a web front end that allows the user to upload files and assists in constructing the metadata file.

Automated analysis is performed by tools (typically simple scripts that wrap standard analysis tools) that consume existing data sets in order to generate new data sets. The metadata file is produced automatically, and incorporates provenance information, including the use of data sets, tools and version.

Data search and *data extraction* by criteria not encoded in the metadata must typically be tailored to specific data types.
Search front ends scan the metadata files, and incorporates all appropriate data in an application-specific search database.

Conclusions

We have described a lightweight framework for generic data storage. This allows scientist to easily submit and make use of data sets, and data managers to easily

incorporate and manage them, as well as implement independent applications that make use of the data store.