PRESERVATION OF ETDs ON NDLTD

Version 1.0

February 3, 2003

Anil Bazaz
Virginia Tech
abazaz@vt.edu

Edward A. Fox
Virginia Tech
fox@vt.edu

Contents

1 Introduction

Theses and dissertations published at a university are important research resources. ETDs (Electronic Theses and Dissertations) are simply the theses and dissertations published in electronic form (e.g., in PDF). Many universities are implementing a requirement that theses and dissertations be submitted in electronic form, thus making it easier for other people to access these works. These ETDs typically are archived on a server at each local university. We have developed a mirroring system which will store additional copies of remote ETDs, and thus will preserve and enhance access to them. The local archive of ETDs will be updated regularly. If someday the university (Publisher) fails to provide access to one of its ETDs or an ETD copy is corrupted, the user will still have access to another copy of ETD. The above system will be used for NDLTD (Networked Digital Library of Theses and Dissertations).

NDLTD is an initiative to encourage the creation of ETDs by student authors, and to make ETDs easily accessible to students via World Wide Web, thus improving graduate education. There are currently over 150 members in NDLTD. Users can browse or search ETDs through the NDLTD website. The NDLTD website also provides a union catalog to search for ETDs.

The Open Archives Initiative (OAI) is dedicated to solving problems of digital library interoperability. OAI has developed a metadata harvesting protocol to support streaming of metadata from one repository to another, ultimately to a provider of user services such as browsing, searching, or annotation. An OAI harvester implements the OAI protocol for metadata harvesting.

We use an OAI harvester to harvest metadata about ETDs and then a simple web crawler is used to get the actual data and store it on a local machine. This ensures that we have a local copy of data even if the publisher of data is somehow unable to provide us with data. Our OAI harvester harvests metadata, which was not harvested since the last time it was run. Hence, updating the mirror site is easily accomplished. This is a very effective scheme, which can be used to mirror any collection of data, provided the collection has an associated OAI server.

2 System Design

2.1 Overview

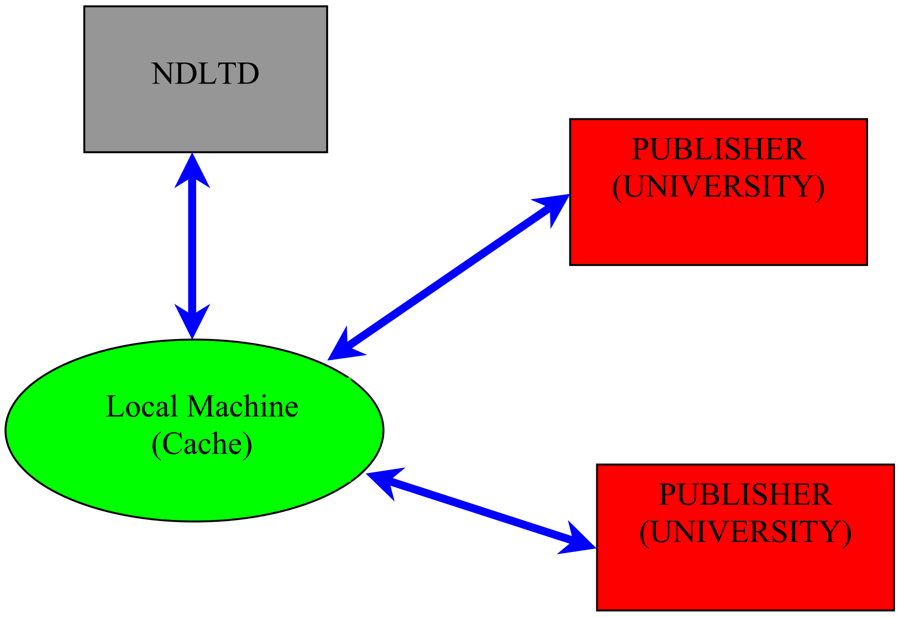

Figure 1. General System Design

Figure 1 shows the general design of the system. NDLTD hosts an OAI union catalog, which is used to search for ETDs. Our OAI harvester is used to harvest the latest metadata from that union catalog. The cache (local machine) stores and parses through the metadata and gets information about an ETD. It then goes to the publisher site and fetches the actual ETD and stores it locally. Harvesting will be done on a weekly basis. This will take into account heavy traffic days, for example the end of semester, and also low traffic days when the number of theses published is low, thus balancing network traffic.

2.2 System Requirements

A cheap PC can be used to run the system. The only requirement being that the PC should have enough storage capacity. Currently a cheap PC with about 120GB of space is approximated at $550 (minimum configuration).

The system is implemented on LINUX and will work with little or no change on UNIX variants. The system was implemented in Perl and so no installation is required. We just have to copy scripts to appropriate directories and we are ready to go. Both LINUX and Perl are free and so no software costs are associated with the system.

2.3 Architecture

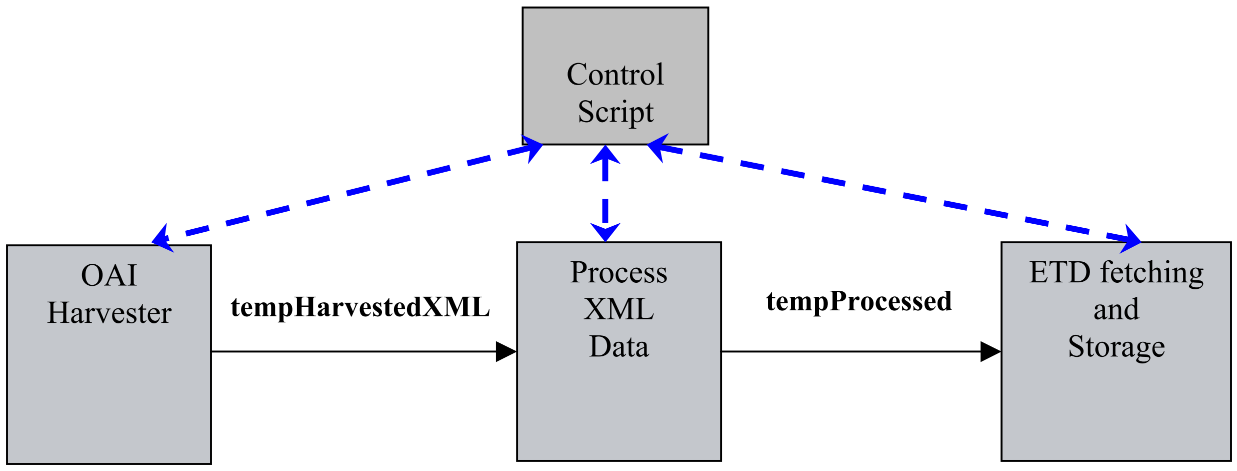

Figure 2. System Architecture

Figure 2 shows the architecture of the system implemented. The system has been implemented with a pipeline architecture. The OAI harvester collects data and stores it in a file named tempHarvestedXML. The file acts as input to the Process XML Data module. This component parses through the file and extracts the etd-identifier, publisher, date of defense, and the URL of the title page for each ETD and puts them into a file named tempProcessed. This file acts as an input to our final stage. In the final stage the data pertaining to each ETD is fetched and stored on the local machine. In addition we have a control script, which is responsible for controlling the execution of the entire system and cleanup operations.

2.4 Local Directory Structure of Cache

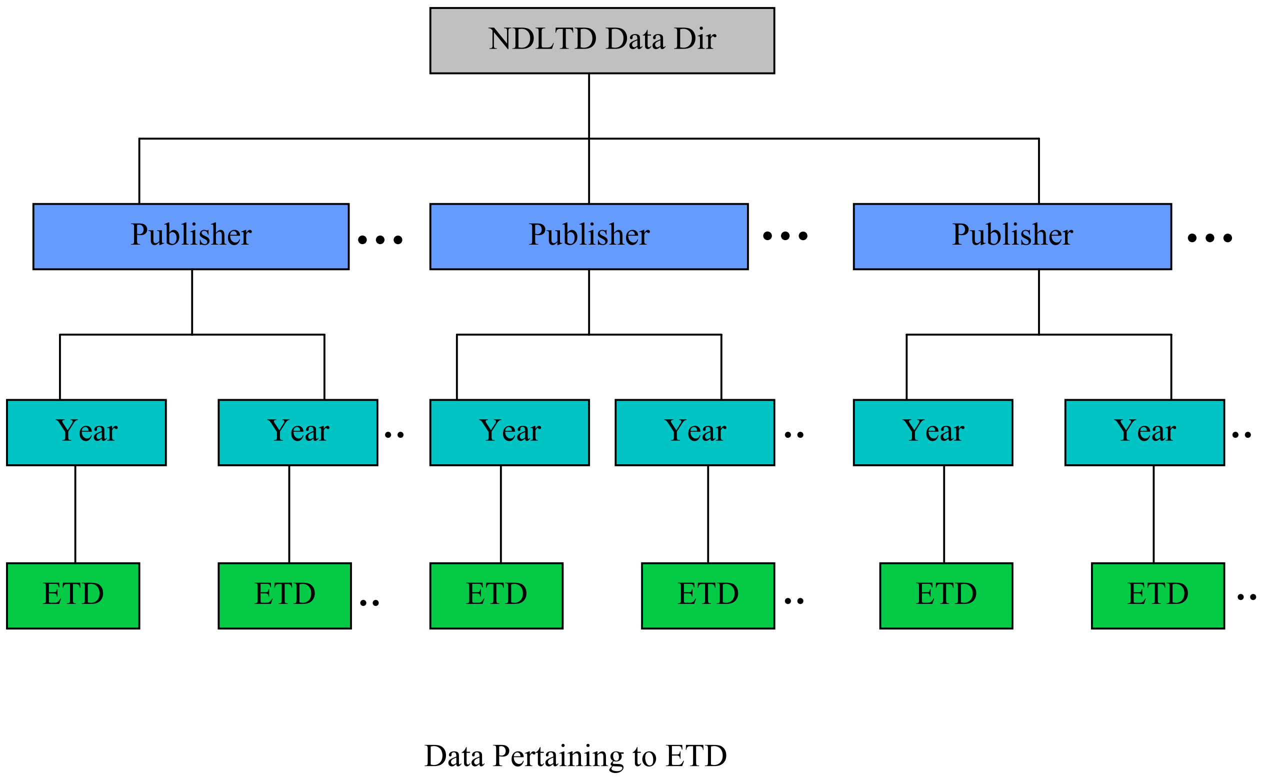

Figure 3. Local Directory Structure

Data fetched from the publisher is stored on the local machine. To store the data in an orderly form we use a local directory structure (Figure 3). We have the root data directory shown in above figure as NDLTD Data Dir. This directory contains subdirectories by the name of each publisher. An example of name of publisher directory would be VirginiaPolytechnicInstituteandStateUniversity (The space has been deliberately removed as the spaces are removed while naming directories.) The publisher directory contains one or more subdirectories named by the year. An example of this would be 1997. The year has subdirectories according to the ETD name. These directories are named according to the individual etd-identifier, a unique identifier assigned to each ETD. In this directory is data pertaining to each ETD. This directory contains an html file, which is the electronic version of a title page of the ETD. This directory also contains all of the other thesis documents pertaining to that ETD. So each ETD will be stored according to its publisher, the year of defense, and its identifier.

2.5 User Guide to Installation and running

1. Create a directory named public_html in your home directory.
2. In public_html create a directory by the name of cgi-bin.
3. The directory structure will be like this *~/public_html/cgi-bin/*
4. cd ~public_html/cgi-bin/
5. Download the component from the ODL website.
   wget http://oai.dlib.vt.edu/odl/software/harvest/Harvest-2.0.tar.gz
6. Decompress the file
   gzip –cd H* | tar –xf –
7. Change to “ODL-Harvest-2.0/Harvest”
   cd ODL-Harvest-2.0/Harvest
8. Run
   ./configure.pl NDLTD
9. Press “a” to add the archive just installed. Answer the questions asked by the configuration script as listed below:
   Archive identifier: NDLTD
   baseURL of the archive: http://oai.dlib.vt.edu/~etdunion/cgi-bin/Union/union.pl
   Harvesting interval: 604800 (One week)
   Harvesting overlap: 1 (default)
   Harvesting granularity: second (default)
   metadataPrefix: oai_dc
   set (leave empty): (default)
10. This completes the installation and configuration of the OAI harvester. Now copy the Perl scripts provided into the Harvest directory.
11. cd /home/
12. mkdir data
13. cd ~/home/<user-id>/public_html/cgi-bin/ODL-Harvest-2.0/Harvest/
14. Start the control script ./control_script
15. This will start the system.
16. You will have to run the control_script every week. You can make a cron job of it or even make it a deamon.

3 Design Rationale

3.1 Scaling Issues

The most important scaling issue is storage and the cost associated with the storage. It is imperative that new members will start participating in NDLTD thus increasing the storage requirements. It is also anticipated that the cost of hard disks will keep on going down at the rate of about 50% per year. If we assume that there will be an increase in members at the rate of 50% per year, still the cost of storage will keep decreasing because of decreasing cost of hard disks. The software itself is scalable and a minimal change in the OAI harvester configuration file (addition of new member set) will enable us to store ETDs from that member locally.

3.2 Articles Stored

The title page and the ETD are stored on the local machine. The ETD itself, along with any other data associated (for example, figures, maps, music, etc.), are stored on the local machine. The title page contains a brief summary of the ETD along with details such as the author, publisher (University), date of defense, etc.

4 Implementation

As discussed above the system is implemented on LINUX in Perl. The system is implemented using a pipeline architecture and has three modules and a control script. We will discuss in detail each of the three modules and the control script in the next sections.

4.1 OAI Harvester

An OAI harvester written in Perl at Virginia Tech has been used in this system. It is quite a simple implementation of an OAI harvester. It has a configuration file associated with it in which the URL of the OAI server, the time intervals at which the harvester is run, the metadata format, and some other options (not specific to this implementation) can be specified. The time interval for this implementation was one week, the site being NDLTD union catalog and the metadata format being oai_dc. The harvester returns metadata of ETDs since the last time it was harvested and stores it in a file named tempHarvestedXML.

4.2 Process XML Data

This component takes as input the file tempHarvestedXML. It parses through the file and extracts etd-identifier, publisher of ETD (University), date of defense, and URL of the title page of ETD. The title page is an introduction page, which contains information about the ETD and also URL/URLs of the theses. It places all the above information in a file name tempProcessed.

4.3 ETD Fetching and Storing

This is the final component of our system. It takes as input the file tempProcessed. It extracts data (etd-identifier, publisher, date, URL) for each ETD. It then creates a directory structure for the ETD. It checks if the directory by the name of publisher exists and if not it creates the publisher directory. It checks if the directory with the name of the year exists and if not it creates the year directory. After this it creates a directory for that ETD by the name of etd-identifier. Then it fetches the title page and stores it in that directory. This title page is again named after the etd-identifier and stored as an HTML page. Next the title page is parsed and URL/URLs of the thesis are collected. Data from the URL/URLs is fetched and stored in the same directory and the files have the same name and format as they have on the NDLTD website. The whole process is repeated for the all of the ETDs harvested.

4.4 Control Script

This script is responsible for controlling the execution of the above modules. The script also cleans up the temporary files after the execution is finished. The script also performs sanity checks for the execution of the above modules.

5 FUTURE WORK

It is proposed that we integrate the above system with LOCKSS (an effort led by Stanford University) to provide a robust method for permanent publishing on the web. The mirroring of data will be done by the above system while LOCKSS will be utilized to maintain the integrity of data. The LCAP protocol (developed by LOCKSS group) will be used to communicate with similar mirror sites around the world to keep checking if data stored is correct or not. Together these will provide an ideal solution for permanent web publishing.

6 References

[1] Vicky Reich & David S. H. Rosenthal, "LOCKSS (Lots Of Copies Keep Stuff Safe)", CEDARS Preservation 2000, York, England at http://www.rlg.org/events/pres-2000/reich.html.

[2] Vicky Reich & David S. H. Rosenthal, “LOCKSS: A Permanent Web Publishing and Access System”, D-Lib Magazine, June 2001 Volume 7 Number 6 at http://www.dlib.org/dlib/june01/reich/06reich.html.

[3] Open Archives Initiative maintains a web page about OAI metadata harvesting protocol at http://www.openarchives.org/OAI/openarchivesprotocol.html.

[4] Web page of Networked Digital Library of theses and dissertations at http://www.ndltd.org.

[5] Digital Library Research Laboratory (Virginia Tech) web page at http://www.dlib.vt.edu/.

[6] The LOCKSS project website at http://lockss.stanford.edu/.