

The Web Laboratory Project

Data Movement and Tracking Report

Asha Balasubramaniam (ab683)

May 16th, Spring 2008

TABLE OF CONTENTS

1. Introduction.....	3
2. Project Overview	4
2.1 Load Component.....	4
2.2 Transfer Component	5
2.3 Tracking Component	6
2.4 Report Component.....	6
2.5 Versioning.....	6
3. Updates to Transfer Component	8
3.1 Resume Capability	8
3.2 Starting the application	9
4. Updates to Tracking Component	11
5. Updates to the Report Component.....	12
6. WebLibraryMetaData	13
7. ExistingDataParser.....	16
8. Proposed Future Work	17
8.1 Automating clean up of backed up files	17
8.2 Download of data.....	17
8.3 Extensions/Modifications to Transfer Component and Report Component.....	17
9. Acknowledgements.....	19
10. Terminology.....	19
11. References.....	20

1. Introduction

The main purpose of the Data Movement and Tracking Team is to transfer crawls that have been archived by the Internet Archive to local storage on the Web lab machine. An automated system has been developed for this purpose which tracks the files belonging to the crawls, downloads them and there is separate back up process then backs the data onto a magnetic tape. The Data Movement and Tracking team switched from the semi – automated to the automated system from the FALL 2007 Semester. This system provides an easy and fast way to transfer and track files from the Internet Archive. The Automated system consists of three components: Load Component, Transfer Component and Report Component. A fourth one known as the Tracking Component is used as an interface to the WebLibraryTracking database which stores information about the tracked crawls on the Scidata1 machine, and it is used both to retrieve and store data into the database. The Automated System is currently located on the Web lab machine (weblab.tc.cornell.edu) under F:\akk34. Last semester a portion of DAT files for crawls EA, EB, EC, ED, EF, and EG were downloaded. This Semester, the focus was more on modifying the various components and this report details the problems faced and approaches taken to solve them and thus modifying the system. In particular there were three particular tasks; first one was to update the report component. Last Semester, a new view to the database that reflected back up information about the ARC and DAT downloads i.e. ArcPath, DatPath, ArcBackupTime and DatBackupTime. To reflect this data in the report component, changes had to be made to the source code. Second task was to add the information of tracked and non – tracked crawls contained in a spreadsheet as a table to a newly created database on the Scidata1 machine named WebLibraryMetaData. Thirdly and the final task was to update the transfer component to leverage the SmartFtp resume capability.

2. Project Overview

The Automated System consists of four components:

- Load component
- Transfer component
- Tracking component
- Report component

2.1 Load Component

The Load component is responsible for populating the Automated System database (WebLibraryTracking) with information about available crawl data. This is the first application that the user should run when using the automated system.

Load Component fulfills these requirements:

- Process the information from the IA to discover available nodes and files for a specific crawl
- Add node and file information to the tracking database
- Update a node or file's information if it has been previously added to the tracking database

Instructions

1. Using command line, go to the folder where the DmtLoadComponent.exe is located – F:\akk34\DmtLoadComponentFA07 on the Web lab machine
2. Execute DmtLoadComponent.exe <CrawlName>

Example: DmtLoadComponent.exe EA

OR

DmtLoadComponent.exe EA EB

This application can be used to load the tracking database with information about multiple crawls.

Note that the Load component is very resource-consuming. The CPU usage may become 100%. It generally takes from one to three days to load information about one crawl, depending on its size.

2.2 Transfer Component

Once information about a crawl has been added to the database, the Transfer component is used to download the actual ARC and DAT files of the crawl from the IA nodes. The Transfer component is responsible for downloading files directly from the Internet Archive servers into a local storage based on the information queried from the Automation System database. It is capable of transferring about 1 TB of data per day and relies upon the SmartFTP library, which provides file transfer functionality. This application is GUI based where the user has to specify the crawl to be downloaded. The user should also specify whether ARC files or DAT files or both should be downloaded. This option comes up on doing a right click once the application starts up.

Specifically the transfer component does the following:

- Query the tracking database for files not yet downloaded for a given crawl
- Download the files from Internet Archive Solo nodes to local storage using FTP
- Update completion and error status of nodes/files in the database as they are transferred

Instructions:

1. Go to F:\akk34\DmtTransferComponentSP08 on the Web lab machine
2. Start DmtTransferComponent.exe
3. Specify the crawl to be downloaded using the drop down box and select whether ARC and/or DAT files have to be downloaded.
4. If only files from certain nodes have to be downloaded use “Process Selected” option else use “Process All” option.

The transfer component starts 9 worker threads in order to process multiple IA nodes in parallel. Each worker thread in turn forks 9 file threads so that each thread can download a file from the given IA node. This multithreaded application results in increased throughput and thus large amounts of data can be downloaded.

For a snapshot of the Transfer Component please refer to “Web Laboratory: Data Tracking and Movement Fall 2007” and for detailed Transfer component specification, refer to [Data Movement and Tracking, Spring 2007 Report](#).

2.3 Tracking Component

The Tracking Component is an interface to the WebLibraryTracking database. It consists of a collection of datasets and is a clean way for C# applications to interface with the database. It also consists of Table Adapters that is used by the ReportComponent to display information about the crawls downloaded and stored in the Tracking database.

Instructions:

The Tracking Component can be accessed using the Source code present in the zipped SourceFA07 and/or SourceSP08 folders on the Web lab machine at F:\akk34

To add new datasets the following procedure should be followed:

1. Open the DmtTrackingComponent in Visual Studio.
2. Right – click on the Project Name and choose the option of “Add New Item”
3. Pick “Dataset (.xsd)” and name it appropriately.
4. Using the Server explorer select the data source that the dataset would be an interface to.

2.4 Report Component

The Report component is used to view the data that is currently stored by the system. It gives information about the crawls being tracked by the system, total crawl size, ARC size, DAT size, IA nodes that the crawl files are present on and other such related information. The source code for this component is hosted in E:\inetpub in the Web lab Machine. Currently the modified report component is present in F:/akk34.

Instructions:

To use it, log into the Web lab machine, open a web browser, and type <http://localhost/DmtReportComponent/>. Note that it usually takes a few seconds to display the page. Occasionally, the request times out. In that case, just re-load the page in the browser.

2.5 Versioning

Since this Semester modifications were made to the various components of the Automated System it was decided to maintain versions of the application on the Web lab

machine. The application that was being used till FALL 2007 Semester has been appended with a suffix of “FA07” and similarly a suffix of “SP08” for Spring 2008. Hence now the Web lab machine contains the following applications in the F:\akk34 folder:

- DmtLoadCompnentFA07
- DmtTransferCompnentFA07
- DmtReportCompnentFA07
- DmtTransferCompnentSP08
- DmtReportCompnentSP08
- SourceSP08

This was for purposes of maintaining a reference to original files along with the modified files. SourceSP08 refers to the modified Visual Studio solution file that contains all the modified components. The Load component has been unmodified this Semester.

3. Updates to Transfer Component

3.1 Resume Capability

The transfer component makes use of the SmartFtp library to download crawls from the Internet Archive to the Web lab Machine. The SmartFtp library supports a “resume” feature that lets the user resume a file download from the point it was left off if the download was interrupted.

Approach:

The first step was to check if the Internet Archive servers supported file transfer resumption using the “IsServerFeature” function of the FTPConnection object. The feature to be checked for this purpose is “ftpServerFeatureREST”. The Internet Archive server does indeed support resuming file transfers and now the task was to incorporate the “resume” capability into the automated system.

The Transfer Component code contains the following two main parts:

NodeProcessingQueue.cs

FileProcessingQueue.cs

As per the specification given in the “Data Tracking and Movement Report – Spring 2006”, the Transfer Component GUI allows the user to enqueue node tasks which are dequeued and processed by node consumers. To process a node task, the node consumer will create a file processing queue and en-queue un-transferred files as file tasks to be downloaded by file consumers. The process of establishing the connection with the IA nodes and starting the file download is implemented in the class *FileProcessingConsumer* which is a part of FileProcessingQueue.cs in the source code.

So far if a file download was interrupted, the handler would delete the incompletely downloaded files from F:/RawData and the files would be downloaded from scratch when the application was re-started.

In order to resume file transfers, it was necessary to know whether the file has been downloaded partially or not. One way to do this was to check, each time a download process began whether the file that has to be downloaded already exists in the web lab machine. This would be the case when a file had been partially downloaded or if it has

been downloaded fully but not been backed up. If this check does not succeed then a new file has to be downloaded.

The next step to be taken when an existing file is identified is to check whether the download of that file has been completed by checking its size against the File Task size.

A File Task identifies a single file (either ARC or DAT) which should be downloaded by a file processing consumer. Once it has been ascertained that file has been partially downloaded, the DownloadFile function of the FtpConnection object is called. This function has the following parameters:

1. Name of the remote file to be downloaded
2. Name of the local file that the download is saved to
3. StartposLo
4. StartposHi

Using the documentation given on SmartFtp website, the last 2 parameters were set to the lowest 32 bits and highest 32 bits respectively of the value of last byte downloaded. This would resume the file transfer from the point to which it was downloaded.

3.2 Starting the application

Problems:

This Semester there were certain issues while trying to execute the Transfer Component.

1. Time out errors while launching the application
2. Only single node thread is forked when using “Process Selected” option in the GUI.

When the application was launched using the DmtTransferComponent.exe, a timeout message appear approximately after 30 – 45 seconds. This error was traced to an SQL command which was named “FillnotDownloaded” in the Tracking Component NodeList dataset.

Approach:

1. The first way to fix the problem was to increase the CommandTimeout value of the SQL command. This had to be done in NodeList.Designer.cs where the FillNotDownloaded function is defined. The CommandTimeout value identifies the

amount of time in seconds that a SQL command would be executed and in event of the command not completing execution, it would time out and gives an error message. The timeout error got solved when this property was set to 200 seconds. However this solution took a lot of time (around 3 minutes just to load the GUI) and hence did not seem feasible.

2. The query named “FillNotDownloaded” took 3 parameters CrawlID, GetArc and GetDat. This query’s purpose was to return the set of IA nodes that contain a particular crawl given by its identifier. In the original code, the latter two parameters were Boolean. Once this query was “configured” again using the option “Configure” in Visual Studio by right clicking on the NodeListTableAdapter , the latter two parameters became “Ansistring”. The code in Program.cs under DmtTransferComponent was accordingly modified to reflect this change in data type. This change solved the timeout problem.

When the DmtTransferComponent is loaded using “Process Selected” option is used in order to select specific files to be downloaded only a single node processing thread is started. This is not the case when “Process All” option is selected which results in normal functioning of the system. This problem needs to be looked into as at this juncture it is not clear if it is indeed a problem or a case of unresponsive nodes.

4. Updates to Tracking Component

In the Fall 2007 Semester, to keep track of the files that have been archived to the magnetic tape, we inserted two additional tables ArcBackup and DatBackup that have the information pertaining to the ARC and DAT files that have been backed up.

We also needed a consolidated collection of the data so that the report component could reflect ARC and DAT information about every crawl. To combine data from separate tables, namely ArcBackup, DatBackup and FileList, we decided to create a view named BackupView2. Now in order to get these values for a particular file, the ReportComponent has to be used. The ReportComponent uses Table adapters in order to pull information from the WebLibraryTracking Database. These table adapters are created using Visual Studio. In order to get information from the Backup view, a new stored procedure has been added to the WebLibraryTracking Database. – GetFileDetailsFromView() . This stored procedure takes in a FileID as input parameter and returns information about the file.

A new dataset called “ModifiedFileList” has been added to the application that draws data using the BackupView2 table adapter. This table adapter contains a method Fill that uses this stored procedure to get file details.

5. Updates to the Report Component

Approach

The report component provides a way to track the crawl downloads. The additional information about a crawl download, the ARC and DAT file information now has to be displayed by the report component. In the report component, the FileDetails.aspx page reflects information about a downloaded file. This aspx page has been modified to include back up information about the crawl. This was done simply by changing the data source of this page from FileListTableAdapter to ModifiedFileListTableAdapter using Visual Studio.

This is done using the following steps:

1. Right clicking the ObjectDataSource in the aspx page
2. Go to Configure Data Source
3. Pick the new data source(In this case
ModifiedFileListTableAdapters.BackupView2TableAdapter)

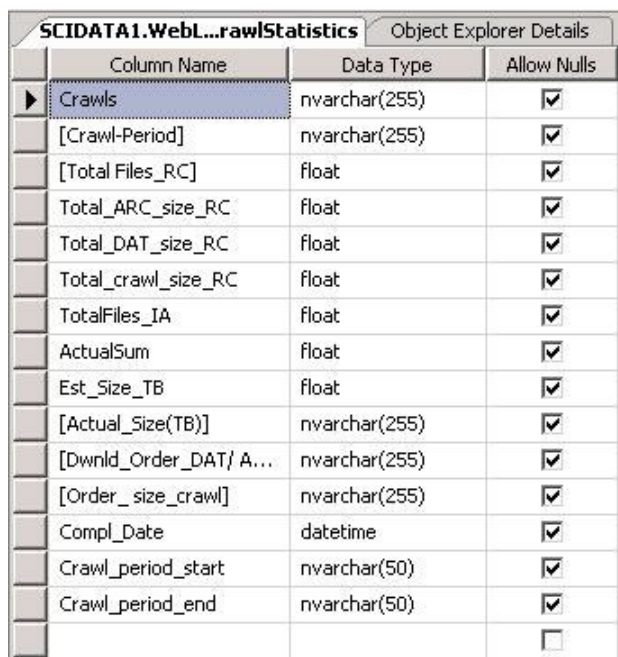
The DmtReportComponentSP08 has been tested on a local machine but in case additional extensions need to be added to this code, the copy has been retained in the F:/akk34 folder and has to be moved into the IIS Inetpub folder on the E drive in the Web lab machine.

6. WebLibraryMetaData

One of the main tasks this semester was to organize information about the downloaded crawls and the ones that are yet to be downloaded. The status of the crawls was being maintained using a spread sheet that recorded the following information. Instead of circulating the information it was decided to put everything in a centralized location and hence WebLibraryMetaData was created. The data from the spreadsheet was split into two tables

1. CrawlStatistics
2. NodeStats

CrawlStatistics gives information about crawls from the spreadsheet and the design is given as follows:



Column Name	Data Type	Allow Nulls
Crawls	nvarchar(255)	<input checked="" type="checkbox"/>
[Crawl-Period]	nvarchar(255)	<input checked="" type="checkbox"/>
[Total Files_RC]	float	<input checked="" type="checkbox"/>
Total_ARC_size_RC	float	<input checked="" type="checkbox"/>
Total_DAT_size_RC	float	<input checked="" type="checkbox"/>
Total_crawl_size_RC	float	<input checked="" type="checkbox"/>
TotalFiles_IA	float	<input checked="" type="checkbox"/>
ActualSum	float	<input checked="" type="checkbox"/>
Est_Size_TB	float	<input checked="" type="checkbox"/>
[Actual_Size(TB)]	nvarchar(255)	<input checked="" type="checkbox"/>
[Dwnld_Order_DAT/ A...	nvarchar(255)	<input checked="" type="checkbox"/>
[Order_size_crawl]	nvarchar(255)	<input checked="" type="checkbox"/>
Compl_Date	datetime	<input checked="" type="checkbox"/>
Crawl_period_start	nvarchar(50)	<input checked="" type="checkbox"/>
Crawl_period_end	nvarchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

This table was created by importing the excel sheet using the import wizard in SQL server 2005.

Following is an explanation of the schema:

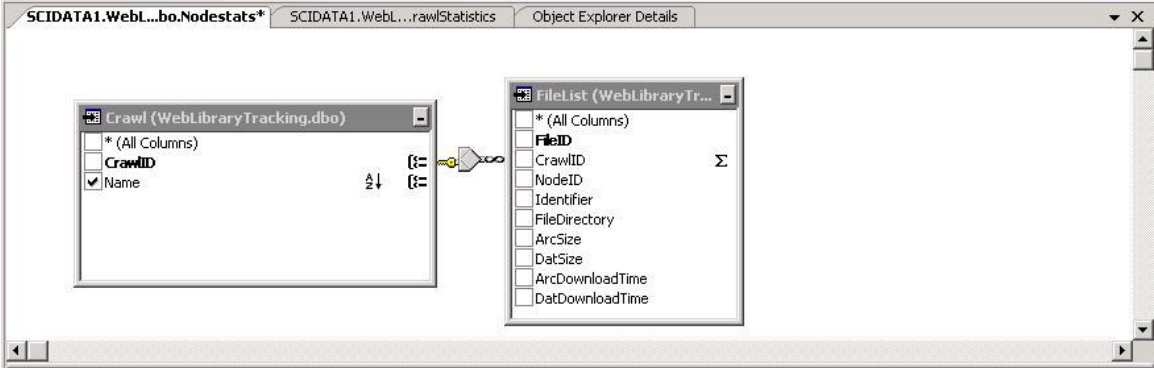
Total Files_RC	This is the count of the total number of files per crawl as recorded in the WebLibraryTracking Database
Total_ARC_size_RC	ARC size of the crawl as per the database
Total_DAT_size_RC	DAT size of the crawl as per the database
Total_crawl_size_RC	Total_ARC_size_RC +Total_DAT_size_RC
TotalFiles_IA	The number of files for the crawl as per the Internet Archive
ActualSum	The total number of files (per crawl) that the Data flow and tracking System could track.
Est_Size_TB	The estimated size of the crawl on the basis of the files data flow system could track. This is computed using the following formula $((\text{ActualSum}/2)*10+(\text{ActualSum}/2)*100)/(1024*1024)$
Actual_Size(TB)	The actual size of the crawl
Compl_Date	Date on which a crawl download is completed
Dwnld Order	The order in which crawls have to be downloaded 1st - Crawls in 2005 2nd - Crawls in 2002 3rd - Crawls in 2003 4th - Crawls in 2004 5th - Crawls in 2001

Crawl_period_start Crawl_period_end	The start and finish date of a crawl determined on the basis of the file identifiers.
--	---

Nodestats:

The purpose of this table was to indicate the number of ARC and DAT files that have to be downloaded for a given crawl. Since this information has to be retrieved using another database i.e. WebLibraryTracking and since the number of ARC or DAT files that have to be downloaded, dynamically changes a view was constructed.

Design of the View



7. ExistingDataParser

The Automated system did not have the records about files that were downloaded during previous semesters using the Semi-Automated System. The records of these files are stored under F-drive on the web lab machine. The ExistingDataParser was run on the following crawl files: DD, DP, DV, DJ, EB and ED. The resulting data was added to the FileList table in the WebLibraryTracking database using bulk insert. The semi-automated system did not give any indication of the ArcDownloadTime and DatDownloadTime. The default.aspx page in the ReportComponent uses the CrawlTabeladapter which computes the number of ARC files downloaded and DAT files downloaded for a given crawl depending on whether the ArcDownloadTime/DatDownloadTime is null or not. Hence even after performing the bulk insert the number of ARC and DAT files downloads in the ReportComponent showed 0 for the crawls mentioned above.

Solution:

The ExistingDataParser was modified so that instead of filling Nulls in the ArcDownloadTime and DatDownloadTime, 0 was inserted so that the ReportComponent could accurately reflect the number of ARC and DAT files per crawl.

8. Proposed Future Work

This Semester the work mainly revolved around modifying the Automated System components. There are many changes that could still be worked upon a few of which are listed as follows:

8.1 Automating clean up of backed up files

Once a file is downloaded to local storage completely it is backed up to a magnetic storage by an automated service.

After the back up is completed a TSM log file is generated in the F:/TSMLogs in the Web lab machine.

After the back up process is complete, BackupDataParser should be run on the TSM log file so that the back up information can be entered into the tracking database.

(Please refer to [The Web Laboratory: Data Movement and Tracking Team Fall 2007 Report](#) December 2007 for detailed steps on using the BackupDataParser.)

After the back up, the user using the automated system has to manually delete the files from the F:/RawData folder. Instead of this step, a service to automate the process of cleaning up the backed up data could be looked into next semester so that sufficient storage can be guaranteed for future downloads and the monotonous work of deleting backed files can be avoided.

8.2 Download of data

Last Semester a part of the DAT data of crawls belonging to the 2005 period i.e. EA, EB, EC, ED, EF, and EG were downloaded. The download of the remaining DAT files and the ARC files belonging to these crawls should be top priority and then the order can be followed. The crawl order can be found here <http://www.weblab.infosci.cornell.edu/tools/getcrawls> in the “Legend” section which is the data pulled from the CrawlStatistics table in the WebLibraryMetadata database.

8.3 Extensions/Modifications to Transfer Component and Report Component

Checking the working of Transfer Component when using “Process Selected” option since only a single thread starts while being used in this Semester.

Also the GUI could be modified to include an “Exit” option when the application wants to be closed after download of some data. Please refer to [Data Movement and Tracking, Spring 2007 Report](#) for detailed specification on the Transfer Component design.

The ReportComponentSP08 has to be moved into the inetpub folder on the E drive of the web lab machine. The current DmtReportComponent contains a file named "App_Web_kkkxvpfs.dll" in the bin folder and it also has a PreCompiledApp file both of which are not present in the DmtReportComponentSP08 folder.

9. Acknowledgements

I would like to thank Professor William Arms and Lucy Walle for their guidance and help. I would also like to thank Wayne Flagg from the Internet Archive for his continuous support. Finally, I would like to thank Manuel Calimlim for helping out with the database problems.

This work is funded in part by National Science Foundation grants CNS-0403340, DUE-0127308, SES-0537606, and IIS 0634677.

10. Terminology

ARC file: A file containing compressed actual web page data.

DAT file: A file containing compressed web page metadata.

Bulk insert: A SQL command used to store a large amount of records to the database.

Crawl: A snapshot of the web for a certain time period.

Flat file: A text file that contains a large amount of records that are to be stored to the database.

Identifier: A string that uniquely identifies a pair of ARC/DAT files. For example, EC_binary1_crawl30.20050306153336

Solo node: An Internet Archive server that contains ARC and DAT data that is not replicated at other nodes.

11. References

1. Asha Balasubramaniam and Dmitriy Shtokman, [The Web Laboratory: Data Movement and Tracking Team Fall 2007 Report](#). December 2007
2. Andrzej Kielbasinski, [Data Movement and Tracking, Spring 2007 Report](#). May 2007
3. Dmitriy Shtokman, [Web Library: Data Movement Spring 2007 Report](#). May 2007
4. Andrzej Kielbasinski, [Data Movement and Tracking](#). December 2006
5. Dmitriy Shtokman, [Web Library: Data Movement Fall 2006 Report](#). December 2006.
6. Sosa, C. B., Jain, P., Shtokman, D., [Web Library: Data Movement Spring 2006 Report](#). May 2006.