

# Data Monitoring and Tracking

-Serena Kohli

-Lipi Sanghi)

ls293@cornell.edu

sk498@cornell.edu

Dept. of Computer Science, Cornell University

16<sup>th</sup> Dec 2005

## Abstract

The WebLab project is a collaborative effort by researchers to create an intelligent archive of Internet web-pages. It involves passing raw web content through a workflow of interacting components that parse and store the data in backend databases. The Data Monitoring and Tracking Unit (DMTU) is designed to track the locations of individual files within this workflow, as well as facilitate interaction between components by attaching flow-relevant metadata to each file. We provide a flexible monitoring and tracking framework that supports a subset of the existing requirements of the Weblab team, and allows for easy addition of new functionality.

We also attempted to identify the parameters required to be tracked to ensure smooth flow of data and the interfaces those are required to support that. We conclude by highlighting the compromises that might be necessary.

## Introduction

The Weblab project was started with the aim of helping researchers study the evolution of the World Wide Web. One can easily appreciate that it is difficult to conduct experiments on content that is spread across the globe and constantly evolving with time. The motivation behind this project is to provide

researchers with a wealth of analysis and easily accessible information to allow them to study the patterns and trends of information change on the internet.

Currently, implemented archival capability for internet data is restricted to verbatim snapshots of static content stored in online repositories, such as the Internet Archive [2]. The Weblab project seeks to offer more intelligent analysis and classification tools to researchers.

A major step in the creation of an intelligent repository of Internet data is the actual transfer of raw data from the 'wild' into a controlled environment, where it is then transformed and classified into richer forms. In the Weblab project, the Internet Archive is used as a canonical source of raw data; web pages are transferred from it into a local Cornell-based replica (referred to as the Cornell Archive) prior to actual processing.

Once the data is in the Cornell Archive, it is retrieved and processed by a multi-component workflow before being stored into a central database. The database can then be mined by researchers through customized APIs and other tools.

As the name suggests, the primary function of Data Monitoring and Tracking unit is to monitor and track all

activity within the WebLab workflow: the actual data files under transfer from Internet archive, as well as internal files created by group members or generated automatically in different stages of the workflow.

Apart from file monitoring and tracking, the DMTU is responsible for keeping track of the disk-space utilization in every stage of the project, the location of any file and exceptions encountered during the processing. The main challenge in architecting DMTU was to come up with the architecture and a schema to store the necessary tracking information.

## The Weblab Workflow

In its current instantiation, the Weblab process involves staged interactions between four components, each of which is implemented and managed by a different group of human operators. We focus on the stages that involve file manipulation.

Following are the important tasks and activities involved:

1. The backing up of the files coming from the Internet archive to the Cornell Theory Center.

The Cornell Archive is physically located in a collection of machines at the Theory Center. This raw-data archive provides the Weblab project with a fast and easily accessible source of input data, and can potentially be used by other research groups in the future. As of now, these files will remain archived for an indefinite amount of time, until storage space is a consideration and

more intelligent decisions need to be taken for selecting raw data to retain.

At this stage our first design decision is taken as we store metadata about the archived files, these archived files are then used by the preload component.

2. The preload component requires a mechanism to be notified that the files are ready to be processed. Also a notification is required for indicating which files have been copied to their working area.

After they have processed the files, we update the tracking database with the log files those have been processed by the Preload component and the output files those have been generated (Parameters of interest are size of the output, identifiers, timestamps and some other information if present). The preload component is not currently logging the output files generated. DMTU group has notified these requirements to the Preload group.

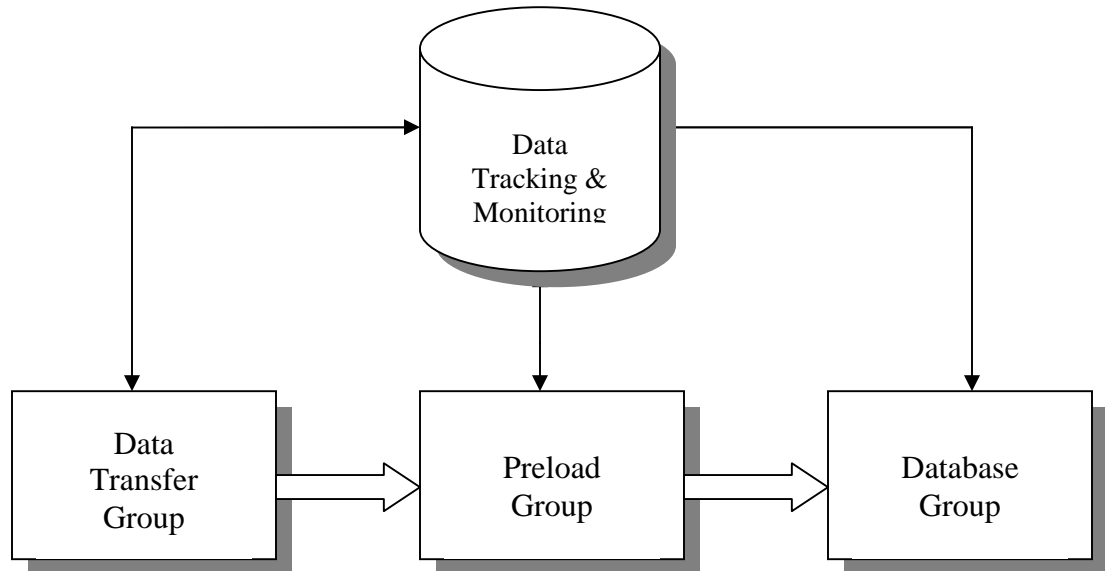
3. The files then generated by the preload group are used by the Database group to load the central database (which is different from the tracking database).

We held discussions with the database loading team on their information acceptance from the preload group, usage and the architecture. The requirement of this group is to indicate to the monitoring system about start as well as completion of loading of a particular table.

This could be done either via some service running in the background or the tracking system may implement it by exposing an interface. As of now we have implemented a record-keeping

system. A notification and control system need to be built on top of it to support other operations.

4. The user-tools group needs to build tool on top of information in central repository for the researchers to use.



Work-flow Architecture

## Current Work

The project started with interviews and meetings with the other members of the project. We meet with leaders of all the component owners and gathered their requirements from tracking database. We decided to build the system iteratively. The current iteration would support record-keeping with the eventual goal of automation of the entire monitoring and tracking unit with the help of this evolving architecture. We designed a schema that would support the current tasks in hand as well as scalable in nature to address future demands.

## Cornell Archive:

Once the files reach Cornell, they are archived using Tivoli Storage Manager (by IBM). This archiving step can be seen as the locking mechanism for the files before they can be used by the pre-load group.

Ruth, the expert in this area, at the Cornell Theory Center and our project leader, helped us with the details of the storage manager and its functioning. At this stage we are logging all the information about the files that have been archived successfully by periodically querying the storage manager and updating the tracking

database with this information. We decided to store the following information about the files:

- File Name
- File Size
- Date and Time when file was backed up
- Management Class
- Active/Inactive
- Preload\_status

The active/inactive flag is maintained by the storage manager to indicate whether the archived file exists on the disk (from where it was sent to back-up) or not. So we decided to use this flag for our tracking purposes as well.

The preload\_status, this flag is put with the intent to clearly differentiate between the read and unread files by the preload component.

### **Preload**

The preload group then takes the files for processing and opens up the .arc and .dat files and extracts the necessary information from them and writes in specified formats.

After many discussions with the component architects we agreed upon a log format which they would be generated to help us in tracking. We mutually agreed to keep the following information:

- File Name
- File Size
- Date and Time when the log was written into
- Processing Time
- Sender of the log information
- Status
- Importance Level
  - § 0: default
  - § 1: information
  - § 2: error
  - § 3: critical

We record the 'status' for every file that is processed by the Preload group as they log when the processing of the file starts as 'processing started' and again when the file has been processed as 'processing ended'.

Sender of the log information is the module within the preload phase which actually parsed the files.

Second part of tracking at this stage comprises of tracking the files that are being generated as a result of parsing the original .arc and .dat files. Currently the preload group is not logging this information and we have notified them of this requirement. Following are the proposed fields that they should log:

- File Name
- File Size
- Date File was Created
- File Type
- Database\_Status

Database\_Status is a flag to indicate whether the database group has taken the preload file and parsed it to fill up the database or not.

### **DataBase**

After discussions and meetings keeping in mind the constraints of the system and the various interactions the database group has with the other modules in the system, we mutually decided to record the following information about the database files:

- Table Written Into
- Size of the Table
- Crawl Id
- Date and Time when Log was written
- Status

The database group takes in the files and builds tables for each crawl separately and the files returned by the preload files are in a format specified by the database group, using the above two information we don't need to track the files separately at this stage, the information that processing for a particular table is complete indicates the files have been successfully processed at this stage. The status field is reset when processing for the file starts and set when the processing is complete.

### Other tasks

**File Location Tracking:** To facilitate tracking of the exact location of a particular file, we decided to maintain two tables, first to keep track of location of the file in the system, and it has the following attributes:

- File Name
- File Location Id

The second table is a look up table where the description for the location of the files, it has the following attributes:

- File Location Id
- Description

We decided to separate the information and make a look-up table such that we can scale the information easily and it gives us all the advantages a normalized database gives.

**Exception Record Keeping:** To facilitate keeping track of all the exceptions that take place, we decided to maintain two tables. The Exception Table which maintains the details of the exceptions and the other is a look up table associated with it, which contains information /description about the type of exception. Following are the attributes for the Exception Table:

- Exception Id

- Exception Type
- Date and Time when Exception took place
- People Informed

Following are the attributes for the Exception LookUp Table:

- Exception Type
- Description

### User Interface

The other task assigned to us was to build a user interface to query the tracking database and display the results to the users, for this purpose we designed the following web-pages:

Page 1: **Main Page:** This is the page that shall be displayed to every user when the user interface is launched, from here the user can navigate and reach the database page, preload page, the backed up files page (Cornell archive), file location tracking page or the exceptions page.

Page 2: **Cornell Archive Page:** This page displays all the information about the files that have been brought in from the Internet Archive and have been backed up at the Cornell archive. The user can perform searches on the basis of the range of dates over which the files were backed up. From this page you can also go to the exceptions page or go back to the main page.

Page 3: **PreLoad Page:** This page displays all the information about the files that have been parsed by the preload group, the user can perform searches on the basis of the range of dates over which the logs were written for the files. From this page you can also go to the exceptions page or go back to the main page.

Page 4: **Database Page:** This page displays all the information about the tables that have been written into by the database group, the user can perform searches on the basis of the range of dates over which the logs were written for the tables. From this page you can also go to the exceptions page or go back to the main page.

Page 5: **File Location Information Page:** This page displays information about the location of the file corresponding to the name of the file. The user can perform searches on the basis of the file name or the file location. From this page you can also go to the file location look up page or go back to the main page.

Page 6: **Exception Information Page:** This page displays information about the exceptions/abnormal instances. The user can perform searches on the basis of the range of dates over which the exceptions could have taken place and exception type.

## Implementation Details

The implementation was done using .NET framework (ASP.Net and C#) and MsSQL. Due to the absence of a web-server for deployment, all the testing was done locally on personnel machines using Web-Matrix (application development tool for ASP.NET with database support) and MSDB (as the database).

The implementation can be divided in two stages, one to parse the log files and update the database and second to generate the web-based reports.

## Results

The architecture and the database for the DMTU are in place. The program was able to parse the given log files and update the database. Also the same database was then queried to display the reports.

## Future work

Further work in this area involves deploying the system on the web-server (once that's made available to us).

We need to automate the entire process of loading the data into the database and then build a notification framework to support all the tasks.

As logs would continue to grow with the time, there could be performance issues unless the non-functional requirement of scalability is given due thought.

The current algorithm needs to be benchmarked for the point where it becomes a "run time" bottleneck on the given hardware. More efficient data transfer and fine tuned algorithms need to continuously evolve along with the growing information storage and processing requirements.

## Conclusion

In this paper, we have presented a flexible monitoring and tracking framework for the Weblab project.

## References

- 1] Web Laboratory Project - <https://gforge.cis.cornell.edu/projects/wri/>  
<http://www.cs.cornell.edu/wya/weblab/>
- 2] The Internet Archive - <http://www.archive.org/>

## **Acknowledgements**

This work is part of the Web Laboratory, which is a joint project of Cornell University and the Internet Archive. Other members of the team are: Professor William Arms, Ruth Mitchell, Lucy Walle, Pavel Dmitriev, Selcuk Aya, Parul Jain, Harsh Tiwari, Dmitriy Shtokman, -Daou Gu Dao, Blajez Kot, Megha Siddavanahalli, Swati Singhal, Wei Guo, Samuel Stern, Chris Sosa, Nicholas Gerner, Nicholas Hamatake and Shantanu Shah. The work is funded in part by National Science Foundation grants 0403340 and 0127308, with equipment support from Unisys, Microsoft, and Dell.