EVENT-BASED MAINTENANCE OF DIGITAL LIBRARY COLLECTIONS

Wendy Osborn
Department of Mathematics and Computer Science, University of Lethbridge, Lethbridge, Alberta, Canada
wendy.osborn@uleth.ca

Keywords: Digital libraries, Greenstone, Event-based collection management.

Abstract: We consider the problem of updating a digital library collection automatically when certain events occur. Existing digital library systems require the user to either re-build manually or to schedule re-building at specified times. Ideally, a collection should be re-built whenever some collection-changing event occurs. We propose the Event Monitor for Greenstone, a tool that continually tests for collection-altering events and re-builds the collection when these events occur. The Event Monitor interacts with the existing scheduling mechanism on the host system, thereby making it a simple yet powerful tool for event-based maintenance.

1 INTRODUCTION

Many applications generate multimedia documents on a periodic basis. One example is a photo-radar program whose purpose is to catch vehicles that break certain traffic laws, such as speeding. A significant number of pictures of vehicle license plates are generated every day. Another example is a medical information system such as CAIRN (Karanikolas and Skourlas, 2000; Karanikolas, 2007), which would be required to be updated continually to keep its content current if it is going to be useful to physicians. In either case, if these multimedia documents are organized into a digital library, the collection needs to up dated regularly to include new documents.

When managing a digital library collection, the owner usually adds new documents to the collection manually. This is reasonable when documents are added infrequency, or the user always has the time and means to access to a computer that allows them to access the digital library. However, when documents are added constantly to a collection, an automated approach is more practical over a manual one. Ideally, a collection should be updated immediately when any changes to the collection are made.

Most existing digital library systems, such as DSpace (Tansley et al., 2006) and Fedora (Lagoze et al., 2006) require that documents be added manually to a collection. Although both software systems provide application programming interfaces (APIs) that could be used to create an automatic updater, programming knowledge is required for using an API. Greenstone (New Zealand Digital Library Project, 2008; Witten et al., 2009) includes a mechanism for scheduling a collection re-build to occur every hour, day or week (Osborn and Fox, 2007). It works by interacting with the Cron scheduler on the host system. Limitations of this approach include the restrictions to re-build at specific intervals. An approach that rebuilds immediately upon any change to the collection would better serve digital library users, especially those requiring real-time updates to their collections.

We present a solution for event-based management of a digital library in Greenstone. The first part involves the design and implementation of an event monitoring module. The Event Monitor runs every minute and tests a collection for any changes in documents, metadata or general configuration. If any changes are found, a collection building script is executed to re-build the collection. The second part involves modifying the Scheduler (Osborn and Fox, 2007) to accept a frequency value of "now" to indicate event-based collection building, and to create the appropriate Cron record. Therefore, the Scheduler will be responsible for setting up the event-based construction of a collection.

2 BACKGROUND

In this section, we provide some necessary background on Greenstone, the Scheduler, and Cron.

2.1 Greenstone

Greenstone (New Zealand Digital Library Project, 2008; Witten et al., 2009) is a software suite for creating custom digital library collections and making them available locally or via the Internet. A digital library collection contains documents of varying formats, including images, PDF files, Word documents, etc. A Greenstone collection can be customized in many ways. For example, a user can specify the appearance of the interface to their collection, how the collection will be accessed, and the types of documents that the collection can contain.

A greenstone collection can be maintained in two ways. The first is by using the import and build commands that are executed from a command prompt. The second is by using the Greenstone Librarian Interface (GLI) (Witten, 2004), which allows users who are more comfortable with using a graphical user interface to have full access to the same functionality.

2.2 Cron

Cron (Nemeth et al., 2007; Vixie, 1994; Schapira, 2004) is a program for users to schedule tasks that will run automatically at a specified time. A task can be one command, or a script containing several commands that are executed in sequence. Initially, Cron was implemented for Unix and Linux platforms, with most systems running Vixie Cron (Vixie, 1994). Mac OS X also runs Vixie Cron. In addition, versions of Cron now exist for Windows platforms, such as Pycron (Schapira, 2004). We summarize the general ideas behind all implementations of Cron that are applicable to our work.

Every minute, Cron reads task configuration files, called crontab files, that contain a record for every task that is scheduled for execution at a specific time. The format of a crontab record is (*min hr dom moy dow user task*), where *min*, *hr*, *dom*, *moy*, and *dow* are the minute, hour, day of month, month of year and day of week, respectively, *user* is the username that the command will run under, and *task* is the command or script that is executed at the specified time.

Two types of crontab include system crontab and user crontab. A system crontab file is mainly for system administration and maintenance tasks. Also, even if all tasks have a specified low-level username, root privileges are required for modifying a system crontab file. A user crontab file can be set up by any user on the system to execute tasks under their own username. Assuming the user has permission to execute the task, no root permissions are required. Therefore, the Greenstone scheduler uses a user crontab file.

```
30  *  *  *  * /collect/pics/gsdl.pl
59 23  *  *  * /usr/bin/cleanup.bash
00  6  *  *  7 /home/someuser/alarm
00  0  1  1  * echo "Happy New Year!"
```

Figure 1: Sample Crontab File.

```
#!/usr/bin/perl
$ENV{'GSDLHOME'}="/home2/gsdl/gsdl";
$ENV{'GSDLOS'}="linux";
$ENV{'GSDLLANG'}="";
$ENV{'PATH'}="/bin:/usr/local/bin:
      /usr/bin:/usr/local/gsdl/bin/script:
      /usr/local/gsdl/bin/linux";
system("import.pl -removeold pics");
system("buildcol.pl -removeold pics");
system("\\rm -r /collect/pics/index/*");
system("mv /collect/pics/building/*
         gsdl/collect/pics/index/");
system("chmod -R 755 /collect/pics/index/*");


00 0 * * * /gsdl/collect/pics/cron.pl
```

Figure 2: Sample Building Script and Crontab Record.

Figure 1 displays a sample user crontab file with 4 tasks that are scheduled for specific times. The first task, /collect/pics/gsdl.pl, is scheduled to be run at 30 minutes past every hour. The second task, /usr/bin/cleanup.bash, is scheduled for execution daily at 11:59pm. The third task, home/someuser/alarm, is scheduled every Sunday at 6:00am. Finally, the fourth task, which echoes "Happy New Year!", is schedule for execution every January 1st at 12:00am.

2.3 The Scheduler

The Scheduler (Osborn and Fox, 2007; Osborn et al., 2008) is a command-line program that is part of the Greenstone software suite. The Scheduler creates a building script for a specific collection and also interacts with Cron on the operating system. The original version of the Scheduler takes as command-line arguments the name of the collection, the import and build commands (and all of their required arguments), and the frequency of execution (hourly, daily or weekly). The output from the Scheduler is a customized Perl script that rebuilds the collection, and modifications to the Cron scheduling service to execute the script at the frequency specified.

For example, suppose we want to schedule a daily build of the collection *pics*. A call to the Scheduler would resemble the following:

```
schedule.pl
    -import "import.pl -removeold pics"
```

```
-build  "buildcol.pl -removeold pics"
-frequency daily
-colname pics
```

Figure 2 shows the resulting build script and *crontab* record for the collection *pics* on Linux.

In addition, a call to the Scheduler can be made through the GLI (Osborn et al., 2008). Figure 4(a) depicts the Options Pane for the Scheduler. Here, the collection owner can select the appropriate option. Then, they can click on Schedule Action for the scheduling to take affect.

3 EVENT-BASED MAINTENANCE

We present a solution for the event-based construction of a digital library collection in Greenstone. The overall process is presented first, followed by all events that require a digital library collection to be re-built. Then, the Event Monitor is presented, followed by required changes to the Scheduler.

3.1 Overall Process

The following sequence of events is adopted for the event-based building of a Greenstone collection. First, an Event Monitor performs the task of detecting the events that require that the collection be re-built. If any of these events are detected, and the collection is not already in the middle of a re-building process, the collection is re-built using an automation script (i.e. cron.pl) that is created specifically for the collection. This sequence of events is set up by the Scheduler to execute every minute.

3.2 Events

Many events require that a collection be rebuilt in order to reflect changes. The events can be categories into documents, metadata, and collection configuration. Each is presented below.

3.2.1 Documents and Metadata

1. **Adding a Document.** When a new document is added to the import folder of a collection, it is not part of the collection until it is re-built.
2. **Deleting a Document.** Similarly, if a document is deleted from the import folder, it is not removed from the collection until it is re-built.
3. **Updating a Document.** Also, if the content of an existing document needs to be updated, this can only occur with the copy in the import folder. Therefore, the collection needs to be re-built to reflect changes in any document.

The same situations apply to user-specified metadata. If any metadata is added to a document, updated for a document or removed from a document, the collection must be rebuilt to reflect the changes. In addition, any indices and browsers that depend on those metadata values must be updated to reflect any changes.

3.2.2 Updated Collection Configuration

The configuration of a Greenstone digital library can be modified in the following ways:

1. **Plugins.** If documents of types currently unknown to the collection are introduced, the corresponding Plugins that are required for processing them must be specified.
2. **Indices and Classifiers.** When working with both indices and classifiers, the collection owner can update the following. The first is the type of index or classifier that will be created. The second is the metadata that is organized for searching or browsing (for example, dc.Title). The third are options for the browsers.
3. **Display Formatting.** In addition, a collection owner can specify how a list of documents that is retrieved via searching or browsing is visualized in a web browser.

The first two options - updates to Plugins and Indices/Classifiers - requires a collection to be re-built.

3.3 Event Detection Module

In this section, we present our solution to event detection in Greenstone. The Event Monitor is written in Perl and runs on Linux, Windows and Mac OS X. Perl was chosen because a Perl script is executable across different platforms and meets the cross-platform requirement of Greenstone.

The Event Monitor requires as input the name of collection to be monitored. It also requires that a cron.pl script exists for the specified collection, and that a Cron task is set up to execute the Event Monitor every minute. The latter two requirements are handled by the Scheduler, and will be presented later.

The events that are detected by the Event Monitor are handled in this order: updates to the collection configuration, updated documents, added documents and deleted documents.

3.3.1 Updated Collection Configuration

The configuration of a Greenstone collection is stored in a collection configuration file (i.e. collect.cfg). The modification timestamp of the collection configuration file is compared to the modification timestamp of the archives source file (i.e. archiveinfo-src.gdb). If the collection configuration file has a modification timestamp that is more recent, then the collection configuration has changed since the last time the collection was built, and therefore the collection needs to be updated. This test is performed first because it is the fastest of all to perform. If the configuration has been updated, the collection can be re-built automatically without having to test for the other events.

We compare modification timestamp of the collection configuration file against the archive source file (i.e. archiveinf-src.gdb) modification timestamp instead of against the system clock (minus one minute) for the following reasons: 1) For the first iteration of the Event Monitor, the collection may be out of date by more than one minute and 2) If a collection takes more than one minute to build, as the case is with very large digital library collections, the resulting collection will be out of date by more than one minute. Therefore all time comparisons are performed versus the archive source file.

3.3.2 Added and Updated Files

Each document in the import folder is evaluated to see if it new or updated at the same time. Therefore, only one pass needs to be made through the import directory. A list of documents that have been imported into the collection is obtained from the archive source file. In addition, the modification timestamp for the archives source file is obtained. This is an indicator of when the collection was last built.

For each document, it is first determined if it is in the list of existing documents. Then, its modification timestamp is evaluated against that for the archive source file. If the document is not in the list of existing documents, or if its timestamp is more recent that that of the archives source file, then the document was added or updated since the last build, and the collection must be re-built.

3.3.3 Deleted Files

When a file is deleted from the import folder, it is no longer possible to use it for testing against any documents lists. However, it is possible to perform a quick test to see if the number of documents is less than the number that have been added to the collection. A difference in values indicates that some documents have been deleted from the import directory. A count of files in the import directory can be obtained during the Add/Update stage. Therefore, only a count of the number of documents in the list of existing documents needs to be obtained to do this comparison.

```
schedule.pl
 -colname pics
 -import "import.pl -removeold pics"
 -build "buildcol.pl -removeold pics"
 -frequency now
```

Figure 3: Scheduler Command for Event-based Collection Building.

3.3.4 Metadata Changes

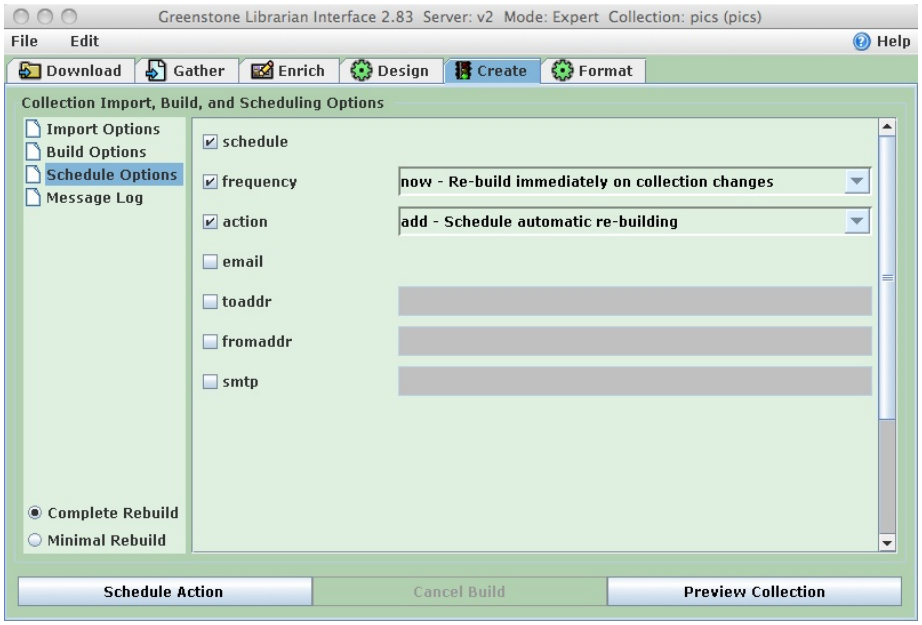
Because any user-specified metadata is stored in a file in the import directory, any additions, updates or deletions of metadata are handled by comparing the modification timestamp of the metadata file against that of the archives source file. Therefore, it is handled the same way as a document update.

3.4 Scheduler Modifications

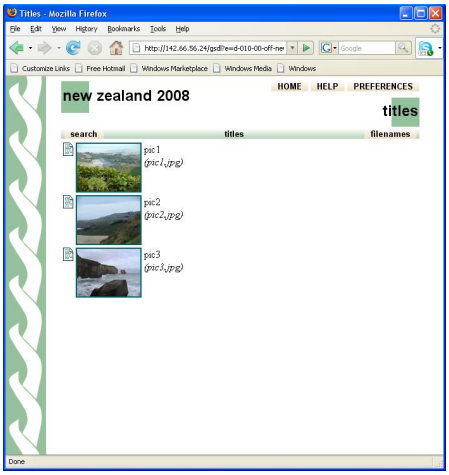
The Event Monitor requires that the collection it is monitoring have an existing automation script and an existing crontab record that schedules the execution of the monitor every minute. The Scheduler creates that appropriate automation script that is tailored to the collection and the needs of its owner. In addition, the Scheduler handles the generation of crontab records for the scheduled build of collections. Therefore, all that needs to be extended in the scheduler to work for event-based collection building are: 1) the specification of a 'now' value for the -frequency option, and 2) for this option, the generation of a crontab record that calls the Event Monitor module. Figure 3 depicts a modified Scheduler command to initiate event-based collection building. Also, Figure 4(a) depicts the GLI with the frequency option value of "now".

4 APPLICATION

We present a simple application of event-based collection building from the GLI. In this scenario, we have a traveler who wants to post pictures of her trip in a Greenstone collection for her friends to view. Instead of waiting until the end of the trip, the traveler wants to post her pictures from her hand-held device, incrementally adding to the collection when necessary. The traveler does not want to worry about accessing the GLI while traveling. Instead, she simply



(a) Scheduling Options Pane.



(b) Build - First Day.

(c) Build - Second Day.

Figure 4: Collection Build Scheduling.

wants to upload the pictures to the import folder of her collection, and have her collection rebuilt immediately after she uploads her pictures. This can be accomplished by setting up an event-based automatic rebuild of the collection of travel photos from the GLI.

Before departing, the traveler runs the GLI and creates a new collection. Then, she selects the Create tab to display the collection creation pane. From here, clicking on Schedule Options will display the available options for setting up a scheduled, automatic collection build of the collection of travel pictures. Figure 4(a) depicts the available scheduling options, which are displayed with default values. Here, the traveler selects schedule, which indicates that she wants to set up a scheduled, automatic build. Also, she selects a frequency of now and an action of add (or, to create a new event-based build).

Next, the traveler clicks on Schedule Action. This will set up the building script for the collection of travel pics, as well as the crontab record that will indicate to Cron that the Event Monitor will continually monitor her collection for new pictures, and automatically update the collection whenever new pictures ar-

rive. The collection is now ready to be re-built while the traveler is away.

On the first day of travel, she uploads three pictures at different times, which are added to the collection immediately upon upload. The updated collection after the upload of the third picture is depicted in Figure 4(b). The next day, the traveler uploads three more pictures at different times. Figure 4(c) depicts the updated collection with the new pictures.

5 CONCLUSIONS

We propose a solution for event-based collection maintenance in Greenstone. The Event Monitor executes every minute and checks for collection-altering events so that the collection can be re-built. Also, this solution takes advantage of the services of the Scheduler, so that the Event Monitor also uses the services of Cron on the host system. Therefore, the Event Monitor is a simple yet powerful edition to Greenstone and will allow collection owners further flexibility in how they manage their collections.

Some future work includes the following. First, although altering display formatting does not require a re-build of a collection to take effect, the collection is re-built in this situation. It would be ideal to not trigger a re-build in this case. Second, if a collection owner wants to not test for events every minute, an ideal approach would be to have an option specifying the number of minutes between event checks. Although the Scheduler ensures that a collection build "in progress" will not be interrupted, some users may feel more comfortable spreading out the event checks anyway. Finally, although preliminary tests were conducted to ensure that the Event Monitor works, a more thorough evaluation of different event scenarios is necessary. In particular, we need to determine whether continuous monitoring will degrade system performance overall.

REFERENCES

Karanikolas, N. N. (2007). Low cost, cross-language and cross-platform information retrieval and documentation tools. *Journal of Computing and Information Technology*, 15(1):71–84.

Karanikolas, N. N. and Skourlas, C. (2000). Computer assisted information resources navigation. *Medical Informatics and the Internet in Medicine*, 25(2):133–146.

Lagoze, C., Payette, S., Shin, E., and Wilper, C. (2006). Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138.

Nemeth, E., Snyder, G., and Hein, T. R. (2007). *Linux Administration Handbook*. Prentice-Hall.

New Zealand Digital Library Project (2008). Greenstone digital library software. Website. Last Visited June 2010. http://www.greenstone.org.

Osborn, W., Bainbridge, D., and Witten, I. H. (2008). A user-oriented approach to scheduling collection building in greenstone. In *Proceedings of the 11th International Conference on Asia-Pacific Digital Libraries (ICADL 2008)*.

Osborn, W. and Fox, S. (2007). Automatic and scheduled maintenance of digital library collections. In *Proceedings of the 2nd IEEE International Conference on Digital Information Management (ICDIM 2007)*.

Schapira, E. (2004). Python cron - great cron for windows. Website. Last visited June 2010. http://sourceforge.net/projects/pycron.

Tansley, R., Bass, M., and Smith, M. (2006). Dspace as an open archival information system: Status and future directions. In *Proc. of the 10th European Conference on Digital Libraries*.

Vixie, P. (1994). Vixie cron for FreeBSD. website. Last visited June 2010. http://www.freebsd.org/cgi/cvsweb.cgi/src/usr.sbin/cron/.

Witten, I. H. (2004). Creating and customizing collections with the Greenstone Librarian Interface. In *Proc. of the Int'l Symp. on Digital Libraries and Knowledge Communities in Networked Information Society*.

Witten, I. H., Bainbridge, D., and Nichols, D. (2009). *How to Build a Digital Library*. Morgan Kaufmann.