GovCom.Org Issue Mapping Project

Final Crawler Technical Report

OneWorld International Tech Team

November 2001

Version	Author	Date	Notes
1	David Heath	19th Nov 2001	
2	Richard Rogers	27th Nov 2001	Sent questions about text to david by email
3			
4			
5			
6			
7			
8			

oneworld.net

1 Table of Contents

2 Introduction

2.1 Purpose of this document

This document is a final technical report from the OneWorld Technical team on the GovComOrg Issue Mapping project. Its aim is to serve several purposes:

- Describe the hardware and software requirements for the system
- System setup and maintenance
- Know problems + issues, and possible solutions
- The technical implementation of the co-link analysis algorithm
- Possible future developments

2.2 Scope

The hardware/software requirements and installation chapters of this document (Chapters 3 and 4) pertain to the Crawler Engine and the Web Front end only.

The remaining chapters on configuration, maintenance and technical evaluation (Chapters 5-9) cover only the Crawler Engine.

This document does not cover the SVG Map Rendering system.

3 System Requirements

As with any computer system, there are basic performance variables, which are influenced in a non-linear way by the amount of processing power and computer resource available to perform the task. With that in mind, the description of hardware given here is more of a suggested specification. Some indication of the performance benefits, tradeoffs and bottlenecks will be given here.

3.1 Hardware

The current system hardware consists of:

- Intel Pentium III 877 MHz (256kb L2 cache) – having a faster processor than this will not make a significant difference to crawl speed. The speed of the internet connection makes a bigger difference.
- 512Mb ram – increasing ram may allow larger crawls to be run.
- 10/100Mbs Ethernet card
- Permanent internet connection with large data transfer allowance (50Gbits/month). Most ISP's will give a default bandwidth allocation of ~ 5Gbits/month. This does not correspond to the *speed* of internet access, but rather the *total data transfer* permitted each month. There will usually be a surcharge for exceeding this. Hotchilli does not artificially restrict the transfer speed to/from the machine, so it is possible to have very fast transfers for a short period of time. This is known as a 'burstable' connection, and is good to have (as long as you know you will not exceed your transfer allowance!).
- 20Gb hard drive

The one world system is currently located with Hotchilli Internet Solutions (http://www.hotchilli.com/). We rent the computer hardware from them.

3.2 Software

3.2.1 RedHat Linux

Version: 7.1 (kernel 2.4.2-2)

URL: http://www.redhat.com/

Installation:

There are many mirror services from which you can download/install redhat. This is only recommended if you have a fast internet connection. Mirrors including ISO CD disk images of the RedHat Linux distribution can be found at http://www.redhat.com/download/mirror.html.

The easiest way to install is to get the CD's. In the UK these can be ordered cheaply from http://www.linuxemporium.co.uk/, or alternatively download the two installation CD Disk images, and write to a cd. There may be other cheap cd places in the Netherlands.

3.2.2 PostGreSQL

URL: http://www.postgresql.org/
Version used: 7.0.2

Installation:

RPM installation packages required.

postgresql-7.0.2-17.i386.rpm
postgresql-devel-7.0.2-17.i386.rpm
postgresql-server-7.0.2-17.i386.rpm

These can be found by searching on http://rpmfind.net/

The system should also work with PostGresQl v7.0.3, which shipped with RedHat 7.1.

3.2.3 FreeTDS

URL: http://www.freetds.org/

Desc: This is the JDBC database driver which allows the crawler to talk to the PostGres database.

This was built from the freetds source which ships with the PostGresQl 7.0.2 source. Rebuild is required to make it work with JDK 1.3 (The JDBC api has changed). Recommend building from the PostGres tar ball, rather than trying to extract it from the RPM. A compiled binary will be supplied with the crawler software, so you should never need to do this, unless you want to use a newer version of the FreeTDS driver.

3.2.4 Sun JDK

Version:

java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1-b24)
Java HotSpot(TM) Client VM (build 1.3.1-b24, mixed mode)

download from http://java.sun.com/j2se/1.3/

Installation:

Just run the install program which will extract an RPM package file which can be installed.

3.2.5 Apache Webserver

Version: 1.3.12-25

Modules: php 4.0.4

Installation:

Install from the rpm, which can be located from http://rpmfind.net/, or ships with standard RedHat distributions.

4 Crawler installation

4.1 Getting started

1. Install all of the required software from section 3.

2. Untar the xxx.tgz file into your home directory. This will set up the default filesystem structure.

3. Set up your path environment variable to include ?? directories

4.2 Database configuration

You need to be logged in as the database super-user, postgres in order to create new users. Do this:

```
$ su postgres                                                    become superuser
$ createdb IssueMaps                                             create db
CREATE DATABASE
$ createuser govcomorg
Shall the new user be allowed to create databases? (y/n) y       create user
Shall the new user be allowed to create more new users? (y/n) y
$ psql IssueMaps
Welcome to psql, the PostgreSQL interactive terminal.            run the psql interactive terminal

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

test=> \i database_8aug2001.sql
CREATE
CREATE                                                           run the database create scripts
CREATE
CREATE
...
                                                                 ...script results
```

4.3 Configuring apache

Apache configuration is beyond the scope of this document. Please consult the voluminous apache documentation available on the internet. Apache should be mostly pre-configured by the rpm installation. However, you will need to make changes for virtual hosts etc.

5 System setup and maintenance

5.1 Crawler parameters file

An example crawl parameters file is given below. The user configurable settings are in the first block, these may be adjusted from time to time to experiment with crawl performance.

The settings in the SystemSettings should be adjusted just once when the system is first set up. Detailed description of the settings is given below:

The tables below give the default values of all attributes.

User Settings		
Setting	**Attributes (name='default')**	**Description**
NumThreads	int='16'	Number of threads used by the crawler. This controls how many simultaneous page requests the crawler can issue. This can be adjusted up to 50 or even 100, but care must be taken not to overload the machine on which the crawler is running
DownloadTimeout	Int='300'	Time in seconds after which an individual http request will be aborted. A request can be aborted even after a download has started, so the request has to complete entirely within the given time.

System Settings		
Setting	**Attributes (name='default')**	**Description**
NumThreads	int='16'	Number of threads used by the crawler. This controls how many simultaneous page requests the crawler can issue. This can be adjusted up to 50 or even 100, but care must be taken not to overload the machine on which the crawler is running
DownloadTimeout	Int='300'	time in seconds after which an individual http request will be

		aborted. A request can be aborted even after a download has started, so the request has to complete entirely within the given time.
DumpTrees	bool='false'	*Do not use this option, it does not work.* Request that all crawled trees are dumped to the console by the crawlSession.
Verbose	bool='false'	Request that crawl/analysis is verbose (lots of console output). This writes a message for each page as it is being downloaded. These messages are normally re-directed to a log file when the crawler is running.
CrawlToDisk	bool='true'	Output all crawling data to disk. Use if anticipating large crawls
DeleteTempFiles	bool='false'	If true then crawler will delete all temporary files generated. False if you want to keep the files, eg. for debugging/further analysis.
ResultTempFileAndNotify	url='http://217.72.169.106/network/crawl_result.php' fileParam='xml_file' resultCodeParam='result_code' resultMessageParam='error_message'	url: gives the url to which result notification is posted The xxxParam attributes give the name of the http parameter to set. These should not be changed. The url should be changed to reflect the location of the web server.
Database	url='jdbc:postgresql://localhost/IssueMaps' user='govcomorg' password='govcomorg'	Control the database url, username and password
CrawlStatsLog	refresh='5'	Frequency in seconds with which the stats log is written to. The stats log records the size of each crawl queue
DisableDatabase	*No parameters*	If present, disables logging of crawler status and command

		success/failure to the database.
TempFilesDirectory	val='/home/govcomorg/tmp'	Path to the temporary files directory
PublicLogFilter	val='FATAL SERIOUS WARNING INFO ASSERTFAILED'	List of log classes to be recorded in the 'public' log which is reported in the crawl results. Possible options: FATAL SERIOUS WARNING INFO DEBUG ASSERTFAILED DEBUG1 DEBUG2 DEBUG3 DEBUG4 DEBUG5
ConsoleLogFilter	val='FATAL SERIOUS WARNING INFO DEBUG ASSERTFAILED DEBUG1 DEBUG2 DEBUG3 DEBUG4 DEBUG5'	List of log classes to be written to the console. Possible options: FATAL SERIOUS WARNING INFO DEBUG ASSERTFAILED DEBUG1 DEBUG2 DEBUG3 DEBUG4 DEBUG5
MaxHosts	val='1000'	If present, restrict the maximum number of hosts to the given number. *This feature is untested*.
MaxPagesPerHost	val='1000'	If present, restrict the maximum number pages to be downloaded per host. *This feature is untested*.

Example file:

```
<?xml version='1.0' ?>
<CrawlParameters>
  <UserSettings>
    <!-- number of threads used by crawler -->
    <NumThreads int='16' />

    <!-- time in seconds after which an http request will be aborted -->
    <DownloadTimeout int='300' />

  </UserSettings>

  <!-- these should only be changed by sysadmin -->
  <SystemSettings>
    <!-- Request that all crawled trees are dumped to the console by the
crawlSession -->
    <DumpTrees bool='false' />

    <!-- Request that crawl/analysis is verbose (lots of console output) -->
    <Verbose bool='false' />
```



```
    <!-- Output all crawling data to disk. Use if anticipating large crawls -->
    <CrawlToDisk bool='true' />

    <!-- If true then crawler will delete all temporary files generated -->
    <DeleteTempFiles bool='false' />

    <!-- controls what crawler does with results -->
    <ResultTempFileAndNotify
                url='http://217.72.169.106/network/crawl_result.php'
                fileParam='xml_file'
                resultCodeParam='result_code'
                resultMessageParam='error_message' />

    <Database url='jdbc:postgresql://localhost/IssueMaps'
          user='govcomorg'
          password='govcomorg'
    />

    <CrawlStatsLog refresh='5' />
    <DisableDatabase />
    <TempFilesDirectory val='/home/govcomorg/tmp' />
    <PublicLogFilter val='FATAL SERIOUS WARNING INFO ASSERTFAILED' />
    <ConsoleLogFilter val='FATAL SERIOUS WARNING INFO DEBUG ASSERTFAILED
DEBUG1 DEBUG2 DEBUG3 DEBUG4 DEBUG5' />

  </SystemSettings>
</CrawlParameters>
```

5.2 Interpreting crawler log files

The crawler log file can provide a helpful means of diagnosing problems. It can also be possible to extract intermediate information from the log files for a crawl which did not complete successfully. The crawler dumps intermediate results, such as co-linkees to the log file at each iteration.

Garbage Collection lines. Each time the garbage collector runs, a line like this is written to the console:

```
[GC 730K->238K(1984K), 0.0018158 secs]
```

This states that the used heap size was reduced from 730K to 238K by the last garbage collection operation. The operation took 0.0018158 secs. The allocated heap after the operation was 1984K .. implying that there is still free space on the heap. When the heap is nearing its maximum size, the garbage collector will run more and more frequently, and the length of processing time will increase. Eventually the garbage collector will not be able to free any more memory and an OutOfMemoryException will be thrown.

Other interesting lines which are worth searching for:

Message: Submitting *nnn* root links

Description: Nnn is the number of root links submitted. Followed by a list of urls used as starting points for this iteration.
Message: [Crawlevent: ????] Description: ??? can be started or stopped
Message: checking to see if finished: not finished nq=128 np=16 more=true Description: nq gives the number of pages remaining in the queue to be downloaded np gives the number of pages currently being processed (downloade or parsed) more = true if there are more pages to be crawled, or false if not
Message: Wed Nov 21 11:30:52 GMT 2001: visited=20 tested=815 queue=414 process=16 Queues: a: 82 b: 0 c: 0 d: 0 e: 9 F: 55 g: 4 h: 0 i: 1 j: 0 k: 0 l: 2 m: 1 n: 0 o: 28 p: 11 q: 2 r: 3 s: 15 t: 0 u: 62 v: 25 w: 11 x: 0 y: 11 z: 6 ab: 62 bb: 4 cb: 12 db: 0 eb: 0 fb: 0 gb: 0 hb: 8 ib: 0 Description: These lines are both written to the stats log. The first line gives overall statistics of number of pages visited (actually downloaded and parsed), links tested (links which have been considered for downloading, but not necessarily downloaded), queue size and number currently being processed. The second line lists the size of queue for each host. The capitalised letter represents the host from which pages will be downloaded next. A round-robin scheduling algorithm is used to choose which host queue will be used next.
Message: [INFO]Total bytes downloaded: 58982050 [INFO]Bytes per second: 122029.89 Description: Gives the total number of bytes downloaded in the previous iteration and the number of bytes per second. Not that this number will overflow if the total number of bytes downloaded exceeds 4GB. This is because a 32 bit integer is used, which can only store integers up to 2^32. This should be fixed to use a BigInteger object. The bytes per second is calculated based on the total bytes divided by time for the crawler to run
Message: [INFO]Performing colink analysis. Mode: 1, priviledgeStartingPoints: true Description: Notify that co-link anlysis is taking place. Mode 0 is by site, mode 1 is by page.
Message: [DEBUG:CRAWLSESSION]reading file: host_skeptics.com.au_8979.xml Description: During colink analysis, each file is listed as it is read in

5.3 Restarting the crawler

The most reliable means to restart a hung crawler is to kill the crawler process using a unix command, and then restart it. There is also a means to request the crawler to terminate, or kill the current job by inserting a command into the database. However, this is not always reliable.

The commands to issue are:

```
# this terminates all java processes running on the machine.
killall java

# if you want to store a copy of the log file for future reference, do something like:
mv nohup.out nohup.out-NN
gzip nohup.out-NN
mv nohup.out-NN.gz ../spool_archives

# Start a new crawler process
nohup ./run.sh &
```

The nohup ('no hangup') command is used to allow the crawler command to keep on running after the user logs of from the machine. Normally all commands issued by a user are terminated when they log off. Issue the man nohup command for more info.

5.4 The run.sh script

The run.sh script is used to start the crawler. There are some settings which may need to be configured in this script. The script serves three purposes:

1. Setup path and other environment variables
2. Setup the java classpath to include the appropriate jar libraries
3. Run the main crawler program using the appropriate java runtime settings.

Taking each of these in turn:

1. You may need to change the JAVAHOME setting if you have installed to a non-default location. The location used is the one used by the jdk rpm.
2. The script automatically adds all jar files in /usr/local/lib/jar and ../jar to the classpath. You can change which paths it scans by altering the export DIR= lines
3. The java command is issued with the following parameters:

-verbosegc requests that the java runtime writes out a message to the console whenever the garbage collector is run. This results in the lines like [GC 730K->238K(1984K), 0.0018158 secs] in the log file.

-Xmx400m specifies that the maximum heap size should be 400 Megabytes. This could be increased for a system that has more ram available. Performance will degrade severely if the heap size exceeds the total ram available in the system

net.oneworld.issuemapper.CrawlerMain is the name of the java class containing the main crawler program.

-x IssueAtlasCrawlParams.xml specifies the xml parameters file to use to set up the crawler (this is passed to the CrawlerMain program).

```
# set env vars
export JAVAHOME=/usr/java/jdk1.3.1/
export PATH=$PATH:$JAVAHOME/bin

# Add jars to classpath
export DIR=/usr/local/lib/jar
pushd $DIR > /dev/null
for file in `ls *.jar`; do
  export CLASSPATH=$CLASSPATH:$DIR/$file;
done
popd > /dev/null

# Add issuemapper jars to classpath
export DIR=`pwd`/../jar
pushd $DIR > /dev/null
for file in `ls *.jar`; do
  export CLASSPATH=$CLASSPATH:$DIR/$file;
done
popd > /dev/null

#
echo $CLASSPATH
java -verbosegc -Xmx400m net.oneworld.issuemapper.CrawlerMain -x
IssueAtlasCrawlParams.xml
```


6 Know problems + issues, and possible solutions

There are currently a number of known problems and issues with the crawler. Most of these are related to the reliability of the crawling process when dealing with unexpectedly large networks.

6.1 Crawler hangs if job too large

A lot of development effort was expended in optimising the crawling process to use less system memory to do its processing. Eventually the system was altered to write all crawl data directly to disk, rather than storing it in memory. This overcame the memory bottleneck during processing, but then exposed a memory bottleneck at the co-link analysis stage.

The co-link analysis process is still done entirely in memory, and will fail if the network is too large to fit into ram (even though only a subset of the actual crawl data makes up the final network). By an informal analysis of crawl log files, it is clear that the crawler is frequently failing at the co-link analysis stage. This is a shame as, by this point the expensive operation of collecting data has completed successfully, only to fail at the analysis stage.

It would be highly desirable to spend some time optimising the colink analysis algorithm to make it more memory efficient. This would hopefully eliminate all crawl failures, and allow even very large networks to be crawled given sufficient processing time. Presently there is a modest upper limit on the size of the network which can be analysed. There is no way for a user to anticipate the size of the final network, and so it is very difficult even to anticipate and avoid excessively large networks.

Most of the other known issues come about due to crawl failures at the co-link analysis stage. Therefore this would be the main problem to fix. Optimisation and testing would take 2-3 programmer weeks.

6.2 Possibility of multiple crawls running simultaneously

Some users have reported that multiple crawls can sometimes be running simultaneously. The system was designed specifically to only allow one crawl to run at a time. However, it may be possible that if a previous crawl did not complete cleanly (either because it was cancelled, or ran out of memory), then it may still remain running. This is a bug related to the unreliable crawl cancellation, and should be fixed at the same time.

6.3 Crawl cancellation unreliable

It seems that the crawl cancellation function does not work reliably. A request to cancel the currently running crawl can be made by inserting an appropriate row into the command/control table (im_crawler). The crawler acts upon this request and aborts the current crawl, but this does not seem to work cleanly, resulting in the

crawler becoming stuck. The unix process must then be killed and re-started. This is a bug which may take 1-2 days to fix and test.

7 The technical implementation of the colink analysis algorithm

7.1 Crawling process

The master control loop of the crawl process is contained in the java class net.oneworld.issuemapper.CrawlSession.

The flow is roughly:

Crawl_roots = initial starting points

While (there are more iterations to do)

Run the crawler using crawl_roots.

Perform co-link analysis on the resulting network.

Calculate the co-linkees which are not on hosts which have already been crawled. These are used as input to the next iteration.

End while

Notice that no host is ever crawled in more than one iteration. This is a simplification of the algorithm, but it makes the crawler significantly less complicated.

7.2 The Crawler

The crawler is based on the Websphinx library from http://www-2.cs.cmu.edu/~rcm/websphinx/

However, it has been significantly re-written, mainly for performance reasons, but also for bug fixing. The HTML parser has been completely stripped down to recognise only the tags relevant to the crawling process. The other significant change was altering the crawler to write all of its results to disk, and discard them from memory at the earliest possible opportunity. The data is recorded in the temp file directory in files named:

Host_*domain.tld_nnnnn*.xml

Where *domain.tld* is the normalised host name and nnnnn is a randomly generated unique number. The files are very simple xml files, which could in theory be mined/analysed in other ways or be re-analysed with different co-link settings. However none of this infrastructure has been built in the current crawler. Only the default one-off co-link analysis is possible.

7.3 Hostname normalisation

For the purposes of comparing the names of hosts, the names are normalised using the following algorithm.

1. Convert the text to lower case
2. If the hostname begins with the string “www.”, then remove that string.

Examples:

www.OneWorld.net becomes “oneworld.net”

slashdot.org becomes “slashdot.org”

```
/**
  * This is the main body of the analysis routine. It processes the file
  * objects in inputFiles.
  * @return a collection of AnalysisSite objects which can be returned
  * from analyse()
  */
 protected Collection analyseMain()
     throws SAXException,
           IOException,
           ParserConfigurationException  {

     // Reads in the log files to our internal data structure, building up
     // the external links from each page/site
     parseFiles();

     // Apply exclusion filters, removing any excluded sites
     applyExclusionFilters();

     // Analyses the external links, and builds up the web of relationships
     // between the pages+sites
     analyseCrossLinks();

     // Filters out all pages/sites which will not remain in the final
     // network, ie those which do not have enough inlinks
     filterNetwork();

     // Categorize all sites in the network into org, net etc.
     categorizeNetwork();

     // Build the collection
     LinkedList l = new LinkedList();
     l.addAll(sites.values());
     return l;
 }
```

Figure 1: the analyseMain function

AnalysisPage, attributes:

Authority: number of hosts which link to this page, multiple links from the same host are not counted.

InwardLinks: list of links to this page, including duplicates from the same host

ExternalLinks: list of links to pages not on the same host as this page.

AnalysisLink:

Knows its target url, and contains a pointer to the page object representing its target.

By-page analysis is done on the basis of authority, rather than ‘inward links’.

8 Possible future developments

As a proof of concept system, the crawler has demonstrated the viability of the co-link analysis approach as a means of learning the shape of known networks, and discovering new ones. The poor reliability of the current crawler is still a hindrance to easy operation by non-technical users. Any future developments should first focus on optimising the colink analysis module and ensuring reliable operation even in the face of an unexpectedly large network. Possible approaches to this and time estimates were given in Section 6.

The ideas listed below should be taken somewhat with a healthy pinch of salt. They are certainly possible, and some organisations already operate crawling systems in these ways. However, these usually involve a significant ($ millions) invest in hardware, software and systems integration expertise.

8.1 Database backed system

The crawler collects a large amount of data, some of which could be re-used or shared between multiple crawl runs, rather than collecting it every time. It may be possible to have a centralised repository of data stored in a database, rather than in xml files on the filesystem. This would record urls, page modifications stamps, date when the item was last crawled. See for example http://valet.webthing.com/how.html for an example of this approach.

There would need to be an automated maintenance system to expire data after a certain amount of time. The database could then be used to as part of the crawl analysis procedure, and also analysed in a number of interesting ways, eg allowing a more interactive querying of the large amounts of data collected by the crawler.

8.2 Distributed crawling

Another idea discussed was the possibility of distributing the crawler engine across multiple hosts, basically allowing for faster crawling. The challenge here would be create the software infrastructure to manage this distributed processing (quite a complex job).

Fortunately, there is an open source project already tackling these issues. The project is called Grub (http://www.grub.org/). It may be possible to adapt the crawler contribute to the grub database, and take advantage of the much larger, more frequently updated collection of data accumulated in that database. The challenge here would be to form and negotiate a mutually beneficial relationship, both on a technical and a business level.

