Brass: A Queueing Manager for Warrick

Frank McCown
Old Dominion University

Amine Benjelloun
Old Dominion University

Michael L. Nelson
Old Dominion University

Original Publication Citation

McCown, F., Benjelloun, A., & Nelson, M. (2007). *Brass: A queueing manager for Warrick.* Paper presented at the 7th International Workshop on Web Archiving and Digital Preservation (IWAW '07), Vancouver, Canada, June 29, 2007.

Brass: A Queueing Manager for Warrick

Frank McCown, Amine Benjelloun and Michael L. Nelson
Old Dominion University
Computer Science Department
Norfolk, Virginia, USA 23529
{fmccown, abenjell, mln}@cs.odu.edu

ABSTRACT

When an individual loses their website and a backup cannot be found, they can download and run Warrick, a web-repository crawler which will recover their lost website by crawling the holdings of the Internet Archive and several search engine caches. Running Warrick locally requires some technical know-how, so we have created an on-line queueing system called Brass which simplifies the task of recovering lost websites. We discuss the technical aspects of reconstructing websites and the implementation of Brass. Our newly developed system allows anyone to recover a lost website with a few mouse clicks and allows us to track which websites the public is most interested in saving.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*Collection*

General Terms

Design, Human Factors

Keywords

digital preservation, search engine caches, web archiving

1. INTRODUCTION

The ephemeral nature of the Web has been well documented over the past decade [4, 9, 14]. Web archives like the Internet Archive (IA) have responded by crawling and archiving as much of the Web as possible. Search engines have also made indexed pages available from their caches although such resources are not kept long-term. These archiving and caching activities have allowed individuals to recover personal and third party websites that would have been permanently lost without these services [15, 20, 29].

To automate the process of recovering lost websites, we created a web-repository crawler named Warrick in 2005 [23]. Warrick recovers lost resources from four web repositories (Internet Archive, Google, MSN and Yahoo) and searches recovered pages for links to other lost resources. The Internet Archive endorses using Warrick [12] since in the past IA employees have spent considerable time recovering websites for individuals who have asked for help.

Warrick has been used by us and others to recover a variety of lost websites. We are currently aware of about 50 specific sites that have been recovered, although our logs show Warrick has been downloaded around 2200 times. Several examples are listed in Table 1, illustrating the wide variety of reasons why websites are lost: hard drive crashes, hacking, accidents, death, etc. None of these events are likely to reduce in occurrence in the future; there will always be a reliance on third-party archives and caches because website backups will not always be available. More detail about some of these recoveries are discussed in [15, 20, 25].

Although we have received numerous emails from individuals who have successfully used Warrick, we have also received numerous requests for help. Many individuals do not have the technical background to install and run Warrick (which requires a properly-configured Perl installation), and some are overwhelmed by the seeming complexity of the command-line interface. Furthermore, Google has stopped allowing users to obtain new Google API keys which Warrick needs to operate [8].

These problems have inspired us to produce an easy-to-use web interface and queuing system for Warrick which we call Brass[1]. Brass allows users to provide websites they would like recovered, and once the sites are recovered, users receive email notifications to pick-up their recovered websites. Brass allows us to track who is using Warrick and what types of websites are being recovered. We also have contact information in case we would like to request additional information about the recoveries; we have used contact information in the past to find interview participants for researching the personal archiving habits of individuals [15].

A simple web interface would have been less difficult to implement except that Warrick is limited in the number of queries it can issue per day, per IP address, as imposed by search engine API restrictions. Therefore Brass must queue jobs to be executed on a number of servers. The distributed nature of Brass required special design considerations. In this paper, we explore the technical processes and considerations of recovering lost websites with Warrick and Brass.

IWAW'07, June 23, 2007, Vancouver, British Columbia, Canada.
.

[1] Warrick is named after a fictional forensic scientist with a penchant for gambling who appears on the CSI television show; Brass is his boss.

Table 1: Sample of Reconstructed Websites

Date	Website description	How it was lost
Oct. 2005	International WWW'06 conference website	Fire destroyed the building housing the web server
Jan. 2006	Kickball organization	Web server's hard drive crashed and no backups were made
Mar. 2006	Christian academic article archive	ISP hosting the site for free discontinued their service
Apr. 2006	Educational site about Roman history	Website owner died, and website eventually ceased operating
Apr. 2006	Fan site of Israeli pop singer Shiri Maimon	Website was hacked, and owner did not have backups
Aug. 2006	Personal website	ISP accidentally deleted the site and did not have a backup
Oct. 2006	Limo company	All ISP's sites were pulled by police because ISP was hosting illegal content
Oct. 2006	US Congressman Mark Foley's websites*	Sites were pulled when Foley resigned over inappropriate conduct
Oct. 2006	Supports sexual assault victims of Darfur*	Unknown
Apr. 2007	Academic law organization in India	Owner accidentally deleted the site
Apr. 2007	Professional technical organization	Web server crashed and backups only partially worked

*Reconstructed by request of the Library of Congress.

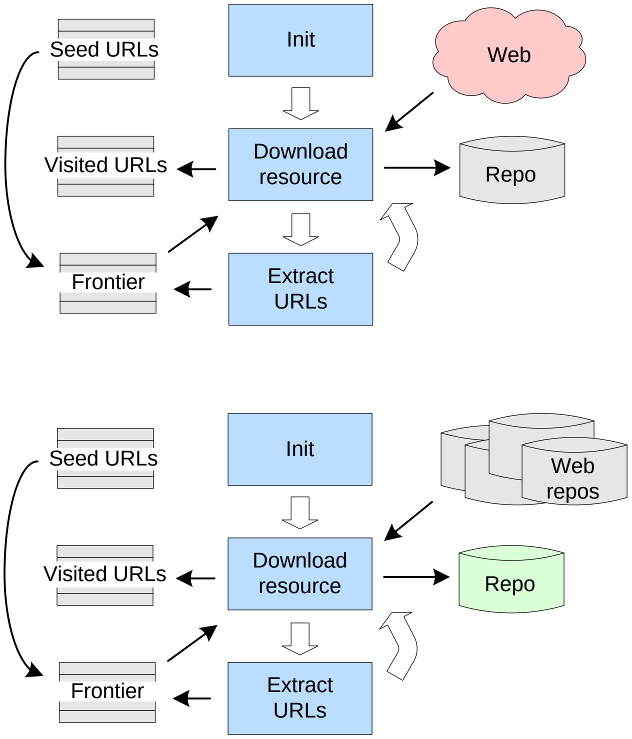


Figure 1: Architecture of a traditional web crawler (top) and web-repository crawler (bottom).

2. CRAWLING WEB-REPOSITORIES

The top of Figure 1 illustrates the architecture of a traditional web crawler [7]. The crawler is initialized with one or more seed URLs which are placed into the crawl frontier. URLs are extracted from the frontier, downloaded from the Web and stored in a local repository. HTML resources (and sometimes other resource types) are searched for additional URLs which are canonicalized and placed on the frontier if they have not yet been visited.

Figure 1 (bottom) shows a similar architecture for a web-repository crawler. A web-repository crawler must also maintain a frontier and list of visited URLs, but instead of downloading from the Web, resources are extracted from web repositories. This requires a web-repository crawler to decide between multiple versions of a resource when more than one are found with the same same URI. The canonical version of a resource may be chosen over a non-canonical version in some cases (e.g., PDF from IA vs. HTMLized version from a search engine cache), or the most recently archived/cached version may be chosen instead of an older version. There are additional URL canonicalizing issues when determining if different URIs from two different repositories are actually pointing to the same resource [20].

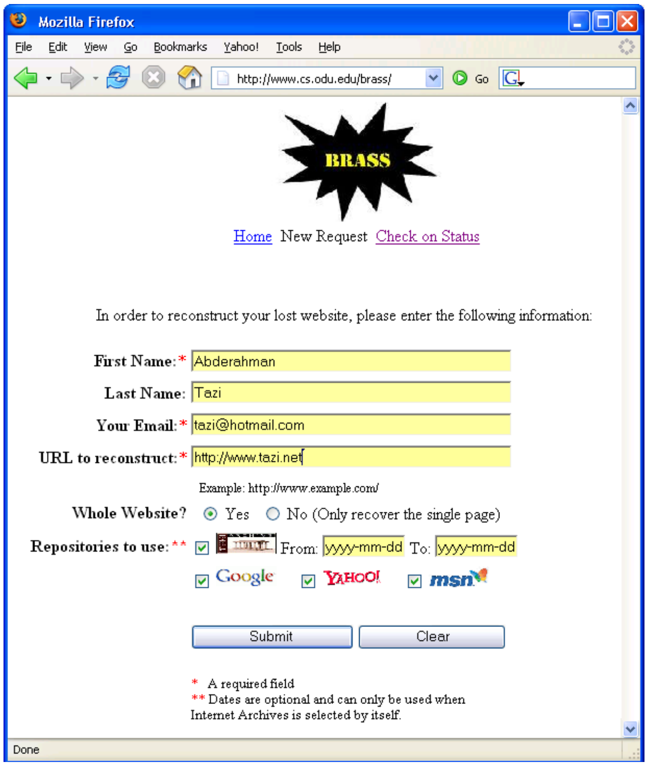
Table 2: Repository Request Methods and Limits

Web repository	Request method	Daily limit
Internet Archive	Page scraping	1000
Google	Google API	1000
Yahoo	Yahoo API	5000
MSN	MSN API	10,000

A web-repository crawler may extract resources from a repository by screen-scraping search result pages and/or by using available APIs. In our implementation of Warrick, we use screen-scraping for accessing IA since no API is available, but we use the APIs for accessing the search engine caches. Although past experimentation has suggested that Google's and Yahoo's APIs are serving from smaller indexes [21], Google explicitly prohibits automated querying of their web interface [10], and Google and Yahoo have temporarily blacklisted our IP addresses in past experiments when they detected we were making automated queries [18].

Just as a web crawler avoids over-burdening a web server by delaying between requests and issuing a limited number of requests in a particular time interval, a web-repository crawler may also limit daily requests to web repositories. The search engine APIs allow only a limited number of queries from distinct IP addresses or from certain keys which Warrick must adhere to (Table 2). Although IA does not publish a request limit, Warrick does not make more than 1000 requests per day per IP address, the same limit used by the Google API. These daily limits make reconstructing large websites a time-intensive process.

Another similarity between web crawlers and web-repository crawlers can be seen in how they find resources from their crawling target. Although web crawlers have traditionally been unable to ask a web server, "What are the set of URLs on your website?" [5], newer methods like the Sitemaps Protocol [28] (supported by Google, MSN and Yahoo) and

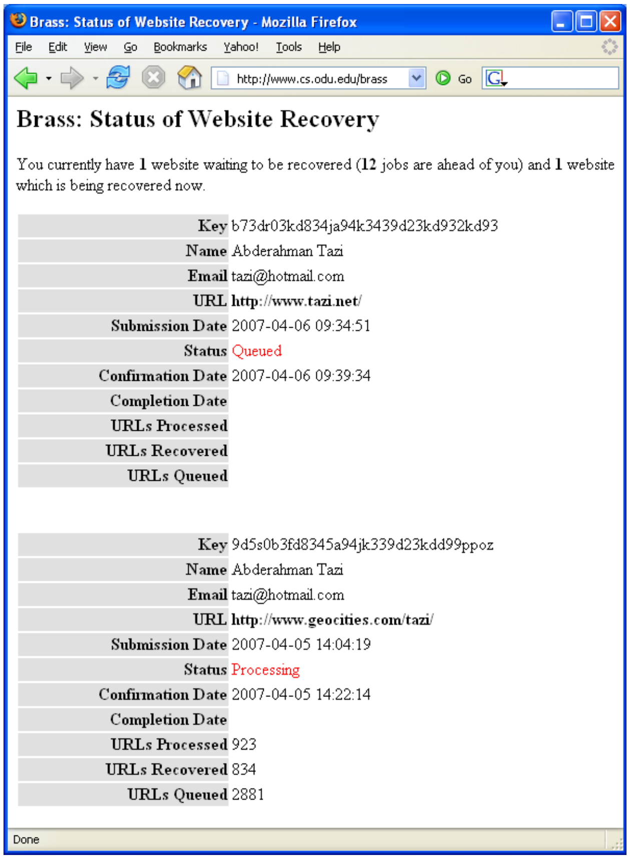

Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://www.cs.odu.edu/brass/
Go
BRASS
Home New Request Check on Status

In order to reconstruct your lost website, please enter the following information:

First Name:*	Abderahman
Last Name:	Tazi
Your Email:*	tazi@hotmail.com
URL to reconstruct:*	http://www.tazi.net
	Example: http://www.example.com/
Whole Website?	● Yes ○ No (Only recover the single page)
Repositories to use:**	☑ From: yyyy-mm-dd To: yyyy-mm-dd
	☑ Google ☑ YAHOO! ☑ msn

* A required field
** Dates are optional and can only be used when Internet Archives is selected by itself.

Figure 2: Screenshot of Brass's submit job interface.


Brass: Status of Website Recovery - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://www.cs.odu.edu/brass
Go

Brass: Status of Website Recovery

You currently have **1** website waiting to be recovered (**12** jobs are ahead of you) and **1** website which is being recovered now.

Key	b73dr03kd834ja94k3439d23kd932kd93
Name	Abderahman Tazi
Email	tazi@hotmail.com
URL	**http://www.tazi.net/**
Submission Date	2007-04-06 09:34:51
Status	Queued
Confirmation Date	2007-04-06 09:39:34
Completion Date	
URLs Processed	
URLs Recovered	
URLs Queued	

Key	9d5s0b3fd8345a94jk339d23kdd99ppoz
Name	Abderahman Tazi
Email	tazi@hotmail.com
URL	**http://www.geocities.com/tazi/**
Submission Date	2007-04-05 14:04:19
Status	Processing
Confirmation Date	2007-04-05 14:22:14
Completion Date	
URLs Processed	923
URLs Recovered	834
URLs Queued	2881

Figure 3: User checking status of two jobs.

mod_oai [26] have been designed to give this functionality to web crawlers. Similarly, most web repositories can be asked to list the set of URLs they have stored for a particular website. We call these types of queries *lister queries* [20]. Lister queries often have limitations; most search engines will not return more than 1000 URLs even when thousands more are cached. When this limit is exceeded, a web-repository crawler must ask the web repository, "Do you have this URL stored?" for each URL since it cannot be sure about its status. This greatly slows down reconstructions of large websites and wastes many queries on un-cached resources.

3. BRASS

When a user wants to recover a lost website with Brass, they are presented with the screen shown in Figure 2. The user's email address is requested so the user can verify the job through email (so automated attempts to start jobs are thwarted) and so we may notify the user when the website has been fully recovered. The user must also provide the base URL of the lost website and indicate which web repositories should be used in the recovery. A date range can be given if IA is selected by itself, otherwise the most recent version of resources from IA is selected by Warrick.

Once the user submits the job, Brass assigns a unique key to the job and sends a job verification email to the user. Once the user replies to the verification email, the job is placed in a queue to be executed once a free machine is available. The job may take several days to process if there are many jobs in front of it or if recovering a very large website. The user may check on the status of submitted jobs at any time (Figure 3). Once the job completes, an email is sent to the user with a link, allowing the user to download the completed reconstruction. Periodic reminders are sent to the user for several weeks if the job is not picked up.

A job moves through four states in Brass:

1. *Pending*: The job has been submitted by the user (Figure 2), but has not yet been confirmed via email. After ten days, pending jobs are removed from the system.

2. *Queued*: The job has been confirmed by the user and is scheduled to run. Although the administrator can reposition jobs in the queue, normally they are scheduled to run in the order they were confirmed.

3. *Processing*: The job is currently running. Due to the query limitations discussed in Section 2, large jobs can remain in this state for several days.

4. *Completed*: The job is finished running and is ready for pick-up by the user (in the form of a gzipped tar file–.tar.gz). An email reminder is sent periodically, and if the job is not picked up in several weeks, it is deleted from the system.

Figure 4 illustrates the overall architecture of Brass. Jobs are stored in an XML file on the web server as shown in the example in Figure 5. Our implementation uses the Apache Tomcat web server [1] which runs on the primary web server and each worker machine. When a job is to be started on

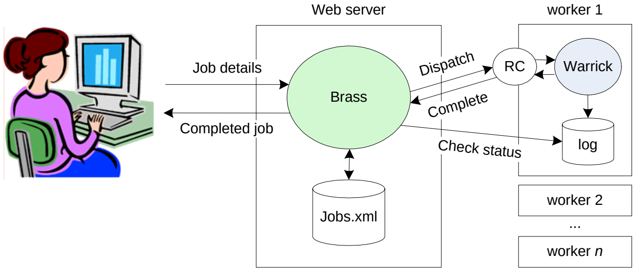


Figure 4: Brass architecture.

```
<job>
  <key>uniquekey</key>
  <fname>John</fname>
  <lname>Smith</lname>
  <email>john@hotmail.com</email>
  <commandLine>--recursive --debug
     --output-file recondir/uniquekey/log.txt
     --target-directory recondir/uniquekey/
     --google-api http://my-lost-website.com/
  </commandLine>
  <submitDate>2007-04-12 09:41:53</submitDate>
  <status>processing</status>
  <reminderToConfirm>1</reminderToConfirm>
  <lastConfReminder>2007-04-12 09:45:26</lastConfReminder>
  <assignDate>2007-04-13</assignDate>
  <processId>124</processId>
  <urlsProcessed>462</urlsProcessed>
  <urlsRecovered>390</urlsRecovered>
  <reminderToPickup>0</reminderToPickup>
  <lastPickupReminder />
  <pickupDate />
  <pickedUp>no</pickedUp>
</job>
```

Figure 5: Job information stored in XML.

a free machine, Brass performs an http GET request with the proper parameters to the Tomcat instance on the target machine. The GET request starts the reconstruction controller (RC), a script which launches Warrick and performs an http GET request back to the primary web server when Warrick completes. This in turn triggers an email to be sent to the user and changes the job's status to complete.

Each instance of Warrick outputs its current status to a log file. For simplicity, Brass and all the worker machines share a common filesystem, so checking on the current status of each Warrick process can be done from the primary web server by examining the log files which all reside in the same networked directory. The log files are accessible to the admin via the web browser for potential trouble-shooting.

The Brass administrative interface (Figure 6) allows the admin to view jobs in their various states. A progress bar is used to show the status of processing jobs (processed URLs / [processed URLs + queued URLs]), and the screen is updated when the page is refreshed. The admin can move jobs in the queue, manually start jobs on specific machines, delete jobs and add new worker machines. A history is kept of all jobs and is accessible through the browser or by downloading an XML file.

4. OTHER WARRICK DEPLOYMENTS

Other methods of deploying Warrick could be utilized in the future. To make running Warrick locally on the client more intuitive, we could produce a graphical user interface, and we could develop a more sophisticated installation program to simplify the installation process. If we wanted to track usage, Warrick could communicate its usage parameters to us at each invocation. Managing usage of Google API keys would be the most challenging aspect since only one Warrick process can use the same key in a 24 hour period. Also it is not a trivial task to create an application which is easy to install on any number of operating systems. Even for programming languages like Perl and Java which are advertised as operating system independent, work must still be done to ensure the proper libraries and runtime environments are installed.

Another method would be to use a web interface, but execute the queries from the client's machine. Java applets, ActiveX controls and Flash are particularly well-suited to such an architecture. In this case the user could download the browser plug-in or client-side application and use a web interface like the one Brass uses. This solution would also require the user to leave their browser window open and unaltered and leave their computer connected to the Internet while reconstructing a website.

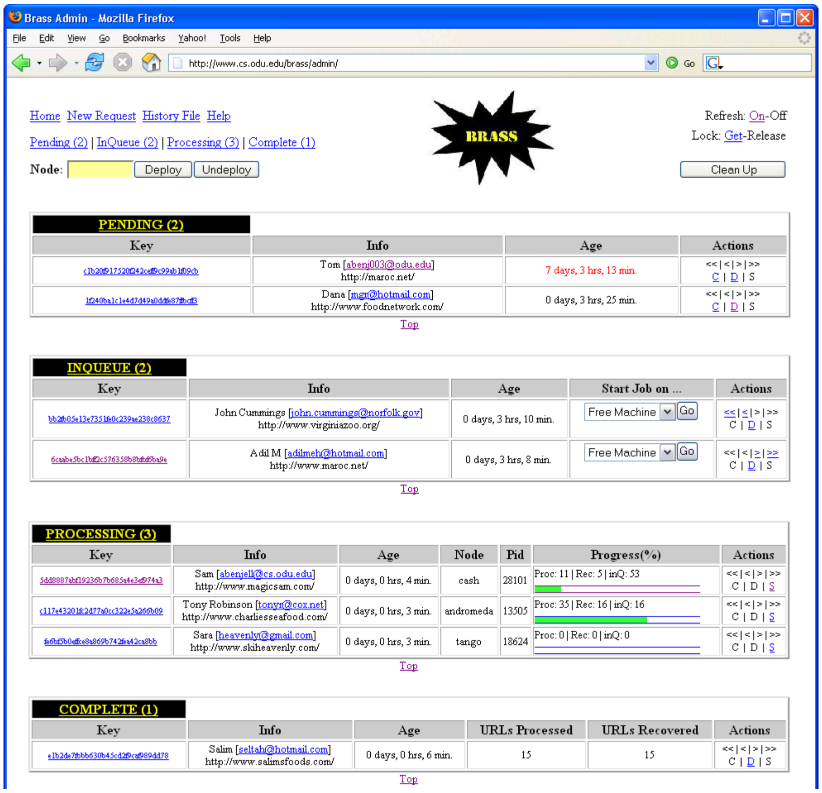
5. RELATED WORK

Web crawling has mainly been studied in the context of search engines (e.g., [2, 6], web characterization (e.g., [3]), web archiving (e.g., [17, 24]) and the Deep Web (e.g., [16, 27]). Our work has focused on using the crawler activity of search engines and web archives for preserving the Web at large and developing a different type of crawler: a web-repository crawler. After our initial experiments with Warrick [23], our work on web-repository crawling has focused on evaluating different crawling policies [20] and characterizing search engine caches [22]. We have also done large-scale experiments monitoring and reconstructing hundreds of websites over a period of 14 weeks to determine the most important factors in reconstructing websites from the Web Infrastructure (the combined efforts of web archives and search engine caches) [19].

A vast amount of work has been done on queueing systems, and we refer the reader to any number of references [11, 13]. Our primary motivation for developing a queueing system is due to the query limit constraints of the search engine APIs. The query limits create a limited resource which we allocate in FIFO order. Without these query limits, we could launch Warrick processes simultaneously on the same machine, and a queueing system would be unnecessary.

6. CONCLUSION

We have described some design considerations in a web-repository crawler and an implementation of a queueing system for Warrick. We have also described some possible future deployment options which would allow Warrick to "charge" queries to users rather than to our servers. We are in the early stages of deploying Brass for public use, and we hope to obtain new insights on what users deem "important," based on usage statistics, in a few months. Based on feedback we have received from Warrick users, Brass will be very warmly received.


Brass Admin - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
Go
Home New Request History File Help
Pending (2) | InQueue (2) | Processing (3) | Complete (1)
BRASS
Deploy
Undeploy
Clean Up
Top

http://www.cs.odu.edu/brass/admin/

Node:

Refresh: On-Off

Lock: Get-Release

PENDING (2)

Key	Info	Age	Actions
c1b20f917520f242ceff9c99ab1f09cb	Tom [abenj003@odu.edu] http://maroc.net/	7 days, 3 hrs, 13 min.	<< | < | > | >> C | D | S
1f240ba1c1e4d7d49a0ddfe87ffbcff3	Dana [mgr@hotmail.com] http://www.foodnetwork.com/	0 days, 3 hrs, 25 min.	<< | < | > | >> C | D | S

INQUEUE (2)

Key	Info	Age	Start Job on ...	Actions
bb2fb05e13e7351fe0c239ae238c8637	John Cummings [john.cummings@norfolk.gov] http://www.virginiazoo.org/	0 days, 3 hrs, 10 min.	Free Machine Go	<< | < | > | >> C | D | S
6caabe5bc1bff2c576358b8bfbf6ba9e	Adil M [adilmeh@hotmail.com] http://www.maroc.net/	0 days, 3 hrs, 8 min.	Free Machine Go	<< | < | > | >> C | D | S

PROCESSING (3)

Key	Info	Age	Node	Pid	Progress(%)	Actions
5dd8887abf19236b7b685a4e3ef974a3	Sam [abenjell@cs.odu.edu] http://www.magicsam.com/	0 days, 0 hrs, 4 min.	cash	28101	Proc: 11 | Rec: 5 | inQ: 53	<< | < | > | >> C | D | S
c117e43201fc2d77a0cc322e5a266b09	Tony Robinson [tonyr@cox.net] http://www.charliesseafood.com/	0 days, 0 hrs, 3 min.	andromeda	13505	Proc: 35 | Rec: 16 | inQ: 16	<< | < | > | >> C | D | S
fe6bf5b0effce8a869b742fea42ca8bb	Sara [heavenly@gmail.com] http://www.skiheavenly.com/	0 days, 0 hrs, 3 min.	tango	18624	Proc: 0 | Rec: 0 | inQ: 0	<< | < | > | >> C | D | S

COMPLETE (1)

Key	Info	Age	URLs Processed	URLs Recovered	Actions
e1b2de7fbbb630b45cd2f9caf989dd78	Salim [seltah@hotmail.com] http://www.salimsfoods.com/	0 days, 0 hrs, 6 min.	15	15	<< | < | > | >> C | D | S

Figure 6: Screenshot of Brass's admin interface.

7. ACKNOWLEDGMENTS

This work is supported in part by NSF Grant IIS-0610841.

8. REFERENCES

[1] Apache Tomcat. http://tomcat.apache.org/.

[2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the Web. *ACM Transactions on Internet Technology (TOIT)*, 1(1):2–43, 2001.

[3] R. Baeza-Yates and B. Poblete. Dynamics of the Chilean web structure. *Computer Networks*, 50(10):1464–1473, July 2006.

[4] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. Sic transit gloria telae: Towards an understanding of the web's decay. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 328–337, 2004.

[5] O. Brandman, J. Cho, H. Garcia-Molina, and N. Shivakumar. Crawler-friendly web servers. *SIGMETRICS Performance Evaluation Review*, 28(2):9–14, 2000.

[6] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of Data*, pages 117–128, 2000.

[7] J. Cho, H. Garcia-Molina, T. Haveliwala, W. Lam, A. Paepcke, S. Raghavan, and G. Wesley. Stanford WebBase components and applications. *ACM Transactions on Internet Technology (TOIT)*, 6(2):153–186, May 2006.

[8] D. Clinton. Beyond the SOAP Search API, Dec. 2006. http://google-code-updates.blogspot.com/2006/12/beyond-soap-search-api.html.

[9] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 669–678, 2003.

[10] Google privacy center: Terms of service, 2006. http://www.google.com/accounts/TOS.

[11] D. Gross and C. Harris. *Fundamentals of queueing theory*. John Wiley & Sons, Inc., New York, NY, 1985.

[12] Internet Archive Web Team. Warrick, a tool for recovering websites, Feb. 2007. http://wa.archive.org/blog/2007/02/22/warrick-a-tool-for-recovering-websites/.

[13] L. Kleinrock. *Queueing Systems, Volume 1: Theory*. Wiley-Interscience, New York, NY, 1975.

[14] W. Koehler. An analysis of web page and web site constancy and permanence. *Journal of the American Society for Information Science*, 50(2):162–180, 1999.

[15] C. Marshall, F. McCown, and M. L. Nelson. Evaluating personal archiving strategies for Internet-based information. In *Proceedings of IS&T Archiving 2007*, pages 151–156, May 2007. arXiv:0704.3647v1.

[16] J. Masanès. Archiving the deep web. In *IWAW '02: Proceedings of the International Web Archiving Workshop*, Sept. 2002.

[17] J. Masanès. Web archiving methods and approaches:

A comparative study. *Library Trends*, 54(1):72–90, 2005.

[18] F. McCown. Yahoo - error 999, June 2006. `http://frankmccown.blogspot.com/2006/06/yahoo-error-999.html`.

[19] F. McCown, N. Diawara, and M. L. Nelson. Factors affecting website reconstruction from the web infrastructure. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital Libraries*, June 2007.

[20] F. McCown and M. L. Nelson. Evaluation of crawling policies for a web-repository crawler. In *HYPERTEXT '06: Proceedings of the 17th ACM conference on Hypertext and Hypermedia*, pages 145–156, 2006.

[21] F. McCown and M. L. Nelson. Agreeing to disagree: Search engines and their public interfaces. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital Libraries*, June 2007.

[22] F. McCown and M. L. Nelson. Characterization of search engine caches. In *Proceedings of IS&T Archiving 2007*, pages 48–52, May 2007. arXiv:cs/0703083v2.

[23] F. McCown, J. A. Smith, M. L. Nelson, and J. Bollen. Lazy preservation: Reconstructing websites by crawling the crawlers. In *WIDM '06: Proceedings from the 8th ACM International Workshop on Web Information and Data Management*, pages 67–74, 2006.

[24] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. An introduction to Heritrix, an archival quality web crawler. In *IWAW '04: Proceedings of the International Web Archiving Workshop*, Sept. 2004.

[25] M. L. Nelson, F. McCown, J. A. Smith, and M. Klein. Using the web infrastructure to preserve web pages. *International Journal on Digital Libraries, Special Issue on Digital Preservation*, 2007. To appear.

[26] M. L. Nelson, J. A. Smith, I. Garcia del Campo, H. Van de Sompel, and X. Liu. Efficient, automatic web resource harvesting. In *WIDM '06: Proceedings from the 8th ACM International Workshop on Web Information and Data Management*, pages 43–50, 2006.

[27] A. Ntoulas, P. Zerfos, and J. Cho. Downloading textual hidden web content through keyword queries. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital Libraries*, pages 100–109, 2005.

[28] Sitemap Protocol. `http://www.sitemaps.org/protocol.php`.

[29] J. Symons. How the Google cache can save your a$$, Dec. 2005. `http://www.smartmoneydaily.com/Business/How-the-Google-Cache-can-Save-You.aspx`.