Overview of the Netarkivet web archiving system

Lars R. Clausen
Statsbiblioteket

May 24, 2006

Abstract

The Netarkivet web archiving system is creating to fulfill our obligation as national archives to collect and preserve Danish internet material. It uses the Internet Archive's Heritrix crawler to harvest data, but surrounding that is a in-house developed system to automate harvests and archive the results. This article presents a rough sketch of the overall system along with some more detail on the module that defines and runs harvests.

1 Introduction

The Netarkivet system was created on the basis of and along with a new legal deposit law that took effect on July 1st, 2005. The law declares material published online as subject to legal deposit similarly to printed or broadcast material. However, unlike for printed material, where publishing is a large enough task that printers can be expected to deposit material to the national archives (a "push" model), web publishing is a trivial task where required deposit would be an undue and unrealistic burden. Instead, the law requires that the two national archives collect material from the web (a "pull" model) for long-term archival.

In recognition of the amount of online material and that we can only vaguely predict which material will be of interest a century from now, the collection is done with a three-pronged approach:

1. a "snapshot" harvest that attempts a broad, automatic sampling of all Danish web pages four times a year,
2. a "selective" harvest of about 80 sites that we estimate to be of particular value, performed with higher frequency and better quality control, and
3. 2-3 "event" harvests per year in connection with important events like elections, royal weddings or major disasters.

The task of collecting, preserving and presenting this material was given to The Royal Library (Det Kongelige Bibliotek, or KB) in Copenhagen and The State and University Library (Statsbiblioteket, or SB) in rhus, the two of which already handle other legal deposit material. Implementation of the system started *when* and took *how many* man-years of work before the first snapshot harvest was started *when* . For a description and analysis of the results of that first harvest, see [1]. This article presents an overview of the full system, along with a more detailed description of the parts used to define and run the harvests.

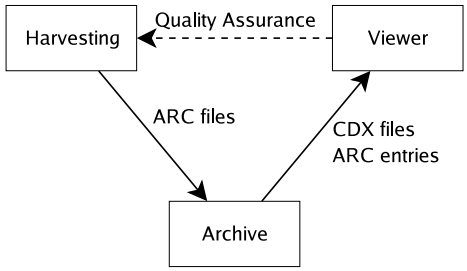

Figure 1: The three main components of the Netarkivet system

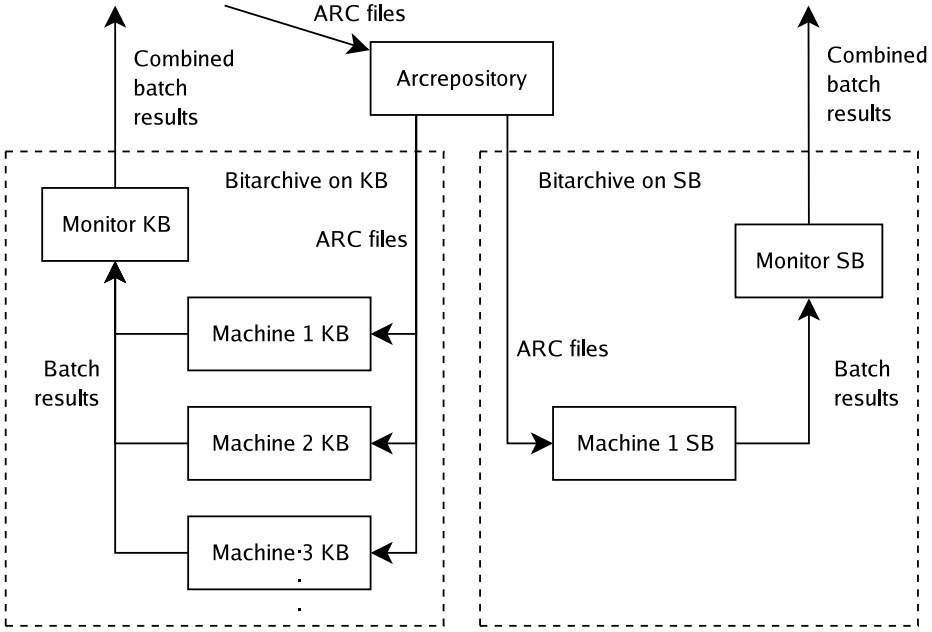

Figure 2: Parts of the archive component

2 Components of the Netarkivet system

Netarkivet is a distributed Java-based system that uses JMS and FTP for communication and file transfer. In our installation, it runs mostly on 8 Linux machines, with 22 Windows machines forming a simple storage cluster for half of the archive. It uses the Internet Archive's Heritrix harvester software for the actual crawls, and stores information on domains and harvest scheduling in an Apache Derby SQL database.

Logically, the Netarkivet system consists of three major components, as shown in figure 1.

The Harvesting component contains the database and web-gui for defining and scheduling harvests, as well as the wrapper program that runs Heritrix and uploads the results to the archive. This component will be further described in section 3.

The Archive component, shown in figure 2, encapsulates a system for redundant storage, where each file is stored in all of one or more archive systems. An "ar-

crepository" module handles the distributed upload, including storing checksums to support bitpreservation. The actual data are stored in one or more bitarchives, each of which may consist of any number of actual machines, thus supporting simple clustering. In order to allow batch jobs such as checksum checks and metadata extraction, each bitarchive has a monitor that can collect and merge replies from the machines in the bitarchive.

The Viewer component component contains a proxy-based viewer that can register unharvested URLs for quality assurance purposes. Due to legal restrictions on the dissemination of sensitive personal data, the data can only be accessed externally by researchers who apply for access for a specific research project. Until these restrictions are relaxed, we concentrate our efforts on collecting and preserving data rather than making the data more easily accessible.

3 The harvest management system

In the harvesting component, archivist can define domains, harvests, schedules and the order.xml templates that go into running Heritrix. From each harvest, a scheduler creates one or more jobs that are transferred to the machines running the heritrix instances. In this section, we describe in more detail the ways harvests can be defined and how the harvests get run.

3.1 Harvest data model

Central to our harvesting model are the concepts of "domain" and "harvest". A domain is, for simplicity, correspond to ¡name¿.¡TLD¿, e.g., `netarkivet.dk` or `kb.dk`, but not `bbc.co.uk`. Each domain contains a number of "seedlists", lists of URLs from which harvesting of this domain can start, as well as one or more "configurations" that describe ways in which the domain can be harvested. A configuration combines one or more seedlists for the domain with a template for how to run Heritrix to harvest the site. These templates allow modification of Heritrix settings outside of those that the harvest definition system knows about. While a few settings, like the maximum number of bytes to download from a domain, must be easily editable and accessible when jobs are created, other settings, like whether to obey robots.txt[1] or submit GET-method forms, needs to be set occasionally but is ignored by the rest of the system outside of Heritrix. This keeps the interface to a manageable complexity while allowing the full power of the Heritrix order.xml file. The settngs managed by the system are merged with the order.xml template when jobs are created.

The definition of a selective harvest specifies a set of configurations on which to base the harvest, as well as a schedule for when and how many times the harvestshall run. To support event harvesting, where typically thousands of URLs are collected to get broad coverage of the event, a harvest definition can be generated from a list of URLs, automatically creating and adding domains and configurations as needed. Selective and event harvests are collectively known as "partial" harvests, since they don't attempt to harvest all known domains.

In contrast to the partial harvests, a snapshot harvest does not require specifying the domains and configurations involved. It automatically includes all domains, using for each domain the configuration marked as default. While each snapshot harvest is run only once, there is an option to base the harvest on a previously completed harvest, including only those domains that were not previously completely harvested. Together with the per-harvest setting for maximum number of bytes to harvest per domain, this allows a "stair-step" approach to harvesting. Starting with

[1]The legal deposit law explicitly allows us to ignore the robots.txt files

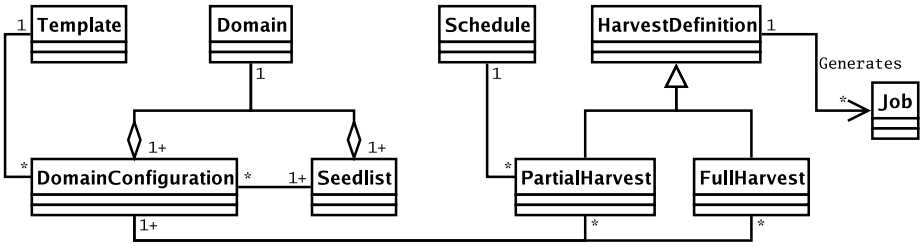

Figure 3: A model of the main classes in the harvest definition system

a low limit to the harvests, such as 10MB, allows us to better balance the domains included in each Heritrix crawl, as well as do a quality assurance check on sites with a large change in size.

The schedules used for harvests are defined separately from the harvests that use them, allowing reuse of common schedules across harvests. A schedule consists of a frequence, a start time, and an end condition. The frequency can be expressed as every N hours, days, weeks or months, either on a specific time within the period or just regularly after the harvest is started. The end condition can be either a date, a given number of runs, or that the harvesty should be repeated indefinitely. However, it is possible to temporarily deactivate a harvest, if for instance problems arise with overharvesting.

A UML diagram for the above model is shown in figure 3.

3.2 Creating and executing a crawl job

Once a minute, the scheduler checks for harvests that are ready to run, and generates one or more jobs from each harvest. The division of a harvest into jobs is dictated partly by the technical requirement that each Heritrix instance can only have one order.xml file[2], and by a load balancing calculation that attempts to ensure a reasonable size of each job. The load balancing must ensure that jobs do not become larger than memory allows, but also attempt to avoid that large and small domains are harvested with the same job. Since Heritrix for politeness has a delay between fetches, a job with many small and a few large domains would cause Heritrix to spend much of its time being idle.

Once a job is created, it is submitted to a JMS queue, and is picked up by exactly one of the harvesters (the exactly-one property is guaranteed by JMS). When the harvesting application recieves a job to crawl, it stops listening for more jobs, writes essential metadata and Heritrix settings to files, and starts up a Heritrix instance. The only addition to the Heritrix code is a QuotaEnforcer that tallies bytes downloaded in a way consistent with out definition of a domain in order to enforce our max+bytes+per+domain limit. While Heritrix runs, the application does nothing except send status messages back to a monitoring system and check whether Heritrix appears to be stuck. If nothing is downloaded for 10 minutes, we stop Heritrix explicitly rather than letting the process run on idle.

After Heritrix stops, whether voluntarily or by request, we generate ARC files containing various metadata that we want to store in the archive. These metadata include the logs and reports generated by Heritrix, the metadata and settings written to files before starting the crawl, as well as a CDX index extracted from the newly generated ARC files. These metadata ARC files are uploaded to the archive

[2]We do not yet support per-domain settings files.

alongside the harvested data. Successfully uploaded files are deleted, while files that fail to upload are put aside for later manual investigation[3]. After upload is complete, a message is sent back to update the database with the status of the job, any significant errors encountered, and some harvest statistics that are used when preparing further jobs. The application then resets and is ready to recieve another job.

4 Conclusion

We have described the system developed and used in Denmark for collecting Danish web material, based on the Internet Archive's Heritrix web crawler.

References

[1] Bjarne Andersen. The DK Domain in Words and Numbers. Technical report, netarkivet.dk, 1 February 2006. snapshot, netarkivet.

[3] In case the application is killed before its data is uploaded, an upload attempt is made on restart if files are found that have not been attempted uploaded.