

Scalability Issues for large scale Web Accessibility Evaluation

Mikael Snaprud, Nils Ulltveit-Moe, Morten Goodwin Olsen, Torben Bach Pedersen*, Christian Thomsen*, Anand B. Pillai+, Terje Gjoesaeter, Helene Unander

Faculty of Technology,
Agder University College

* University Of Aalborg +Spikesource

To enable large scale benchmarking of barriers encountered on web sites, a measurement machinery for a prototype European Internet Accessibility Observatory (EIAO) has been designed.

The first prototype Observatory (release 1.0) demonstrates assessments covering 145 web sites, using a singlethreaded crawler. This first release has focused on correct operation, and not so much on performance. This paper will discuss the experiences made so far, and outline possible ways of addressing performance bottlenecks to scale up for large scale evaluation. The final version of the prototype Observatory will monthly publish updated measurements from 10.000 web sites.

The main elements of the first release of the Observatory are briefly discussed below. They consist of a crawler, Web Accessibility Metrics modules that perform accessibility assessments, and a data warehouse, that will be used to support aggregation and presentation of data from the large scale assessments.

- The key component in the Observatory architecture is the web crawler HarvestMan. HarvestMan was chosen because it is a good and mature web crawler that was easy to adapt to the system. HarvestMan has been integrated with the other components of the Observatory and is essential for the EIAO Observatory.
- A Web Accessibility Metric (WAM) is a formal rule specifying how to make a statement about accessibility barriers of a given Web resource. The project has implemented a WAM container that builds on the Relaxed HTML validator, that uses a Schematron based approach for accessibility assessments.
- The RDF repository consists of two triplestore databases. One is used to store Evaluation and Report Language (EARL) reports that describes the accessibility assessment results from the WAM containers, and the other is used for storing the repository of URLs for the crawler, and other configuration and state information.
- A data warehouse is used in the architecture to support storage of large amounts of data and support flexible analysis from a planned on-line user interface. We also expect the hierarchical design of the databased schema in a data warehouse to facilitate analysis of the data.

Each step in the production line will take some time. The crawler needs to sample simulated use scenarios by performing random walks on the web site. This process involves time for downloading

and parsing web pages, and updating the in-link degree (n) of the different URLs that are visited. All downloaded web pages will be assessed by a set of Web Accessibility Metrics modules that will return EARL reports, that are stored in the RDF repository. Profiling during the test phase has shown that there are several possibilities for improvements and parallelisation, both on crawler, database and WAM level, and this paper will discuss the approaches that seem most efficient for scaling up the observatory to the required size.

Several approaches to attack the bottlenecks will be explored, like distribution, clustering, threading, asynchronous event handling and possible optimisations on underlying database structures for RDF handling.

Distribution and clustering

It is possible to subdivide the set of web sites between a set of web crawlers. This solution would require changes to the ETL process, and we would have to implement some means to synchronise all the distributed crawlers. Storing data in the data warehouse is currently much faster than loading data from the triplestore, so the data warehouse should have capacity to handle a set of crawlers. An appealing feature with this approach, is that it does not require any substantial redesign of the existing system to work. It would, however, benefit from faster crawler/assessment units. An other possible approach is to use an Operating System virtualisation software such as Xen hypervisor allowing multiple instances of the Observatory to run on one machine.

Threading

The HarvestMan crawler is multithreaded, however, some of the extensions that has been written for EIAO are not yet thread safe, so the crawler is limited to two threads. The observatory samples key use scenarios, which are simulated use scenarios using the random walk algorithm. These scenarios are generated as the crawler "reads" the web pages, so that key use scenarios in itself are not possible to parallelise with the current approach. However it should in be possible to run a set of threads, where each thread works on sampling a key use scenario from a given path in the random walk.

Asynchronous event handling

The WAM scheduling is currently sequential, so that each WAM of the set of WAMs to be executed is called in turn. This causes an unnecessary processing delay, because each WAM runs in parallel, so an optimisation would be to add support for asynchronous call handling, so that all WAMs could be called at the same time. The results could then be stored in the RDF store as they became available, and the system would need to implement a barrier (synchronisation point) that ensured that the sampling progressed only after all WAMs had successfully stored the EARL results from the WAMs.

Optimisations on underlying database structures for RDF handling

Currently more than 99% of the ETL's running time is spent on reading data from the RDF repositories, the so-called "extract part" of the ETL. The problem seems to increase as the size of the data in the RDF repositories increases. Currently, the RDF triplestore for EARL reports holds more 67,8 millions triples. The used triplestore, 3store, was designed to handle at least 20 millions triples. 3store uses four tables in an underlying MySQL database to store its data. In one of these tables, a row exists for each triple stored. The problems seems to be that the underlying database structure is too generic and not sufficiently optimized for the large, domain-specific queries that we perform. In the paper, we outline possible solutions to this.

The paper will also address the challenges with the vast amount of storage that is needed for the data

produced by the Observatory, and other potential limitations like network bandwidth and processing needs for web pages and EARL reports.