

Versus: A Web Repository

Daniel Gomes João P. Campos Mário J. Silva

XLDB Research Group
Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749-016 Lisboa, Portugal
[dcg, jcampos, mjs]@di.fc.ul.pt

1 Introduction

The Web is a great personal enhancement tool, but the amount of data available is so vast that its true potential can only be harnessed with applications specialized in aiding users to find, sort, filter, summarize and mine this data. With today's limitations, applications wouldn't be able to solve user queries based on a vast part of the Web in useful time, because it would take them too long to download data.

Pre-fetching the information and storing it, would be a reasonable solution. Getting a copy of all the needed information is very expensive (both on time and bandwidth usage), but saved data can be processed for several applications and users, allowing for reuse.

Our work is focused on the development of a Web repository named Versus, that supports version management, extensible object meta-data, parallel operation and load balancing. Version management allows applications to save the time dimension of data, as well as to save storage space. Extensible object meta-data allows applications to take full advantage of the information stored in Versus. Parallel processing and load balancing allow Web applications to increase their processing capabilities over massive amounts of data. Versus is being developed as a repository for a Web search engine [12], but it is designed as a reusable software component, that could be useful to many modern Web applications in domains such as a data warehousing or GRID computing [4].

2 Functions and requirements

In this paper we consider a Web application (or simply an application), as a Versus client with the ability of executing a task through parallel data

processing. Therefore each application should be composed by a group of independent threads.

Versus enables Web applications to perform the following functions:

Retrieval of large quantities of data from the Web. Applications retrieving and saving data are usually built tightly coupled with the storage system used. Hence, the storage framework should be highly scalable, allowing the distribution of the load among several threads.

Manage meta-data about Web resources. Most applications manipulating Web data require both the documents retrieved from the Web and the meta-data available about these documents. The storage system must provide methods for storing and retrieving these meta-data elements along with the documents.

Save historic data. History may be relevant. Some Web applications might be interested in looking at how a portion of the Web was in the past. The storage must provide access methods enabling user applications to specify what they want to see in respect to time.

To achieve this features we identified the following major requirements for Versus:

- Support for the storage of large amounts of data, reusing and compressing contents.
- Provide support for applications to perform efficient parallel processing of distributed data available on the Web, through partitioning of data into disjoint units, which can be read updated and integrated separately and concurrently;
- Provide extensible meta-data management;
- Enable views on past states of the repository's data, providing a time dimension on stored data;
- Support transitory states of objects in the repository, enabling applications to decide when to make them permanent.

3 Data Model

Figure 1 shows the UML class model of the data handled in a Versus repository. This class model is very general and enables the management of several kinds of saved data. Next we'll describe the classes that compose this model and map them to their meaning in our Web repository:

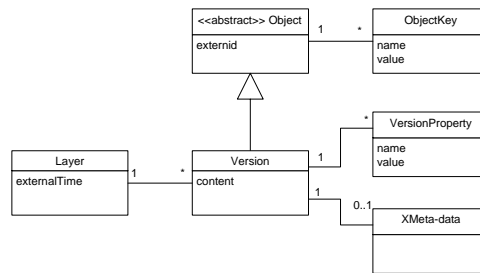


Figure 1: Class model for the data handled by the repository.

- An *object* is an abstraction of a reference to a Web object. In Versus an object represents a reference to a resource on the Web, such as an URL;
- A *version* is a snapshot of an object at a given instant in time. For instance, Versus represents as a version a downloaded HTML document;
- A *layer* represents a logical time in the repository. Only one version of an object can exist in each layer. As an example, our search engine associates a layer to each crawl of the Web;
- Each object has an associated property list *objectKeys*. An application can use the property list to identify subsets of objects to process. In Versus we use an host's URL hash code, as objectKey to refer to all the objects hosted in the same server;
- Each version has an associated property list of *versionProperties*, which is used in Versus to store information such as the download date of a Web page, it's MIME type, size or the status code returned by the server when it was accessed;
- *XMeta-Data* is a container for XML data associated with each version. In our search engine we use it to store anchor texts and the links among HTML documents.

4 Operational Model

Versus is a repository especially useful for applications that can profit from parallel processing of Web data. In these applications, the information stored can be partitioned into disjoint subsets, which can be processed with a certain degree of independence. Applications must supply a function to partition the data into working units, and a function to reconcile conflicting data generated within different units. We assume that the performance

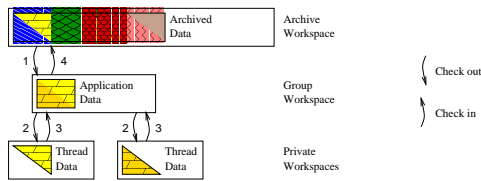


Figure 2: Versus supports three classes of workspaces: archive, group and private workspaces. The figure depicts an archive workspace holding a data set partitioned in several subsets. Applications check-out to the group workspace only the data sets they will use. Threads concurrently check-out subsets of the data, process them, and check them back in.

overhead introduced by synchronization of the non independent parts of the information is largely compensated by the parallel processing.

Data processing is made in a three tier model, each represented by a *workspace*. Workspaces are well bounded and independent environments where application threads can process subsets of data. We define three kinds of workspaces: *private*, *group* and *archive*.

Private workspace: provides local storage and fast access to data by application threads. Private workspaces are independent of one another, and may reside in different machines.

Group workspace: integrates partial results generated by the application threads in private workspaces. Group workspaces maintain a shared view of the data common to all application threads.

Archive workspace: stores data permanently. The archive workspaces keep version history and are able to reconstruct earlier views of data.

Data is partitioned in pieces called *working units*, which are passed from one workspace to another via *check-out* and *check-in* operations. When an application checks-out a working unit, the corresponding data is locked in the source workspace and becomes available only in the destination workspace. When the application performs the check-in, all the data in the workspace is moved back. Versus helps in resolving conflicts between versions and eliminating duplication of contents. Finally, the lock is released. The execution of these operations has the following steps (see figure 2):

1. When an application is started, it creates a new group workspace, checking-out data it will need from the archive workspace;
2. The application then forks n parallel threads. Each one of the threads starts its own private workspace and checks out one of the working units;

3. When finished with one working unit, the thread checks it into the group workspace and restarts with another working unit;
4. Before the application finishes, the results in the group workspace are checked-in into the archive workspace.

5 Proof of Concept

As a proof of concept, we will present in the full presentation the results obtained with a typical usage scenario, provided through the implementation of a prototype of Versus and of a distributed Web crawler, implemented as a Versus application. A distributed Web crawler is an application with high data interaction. Each thread, running on a separate processor, is responsible for collecting documents from certain parts of the Web; in the end, the crawler delivers an archive with the collected documents. The prototypes were used in the building of a new search engine for the Portuguese Web.

6 Related Work

We found systems and research work that inspired our design in Engineering database systems [9], Web-based Distributed Authoring and Versioning (WebDAV)[8], WebBase [6], AIDE [3], WebGUIDE [2], Internet Archive [7], Tamino XML [11], Microsoft SQL Server 2000 [10], Oracle 9i [11], Mercator [5] and Microsoft Repository [1]. These will be discussed and compared against Versus in the full presentation.

7 Conclusions and Future Work

This paper presented Versus a Web data repository, supporting versioning, parallel operation and meta-data management. Versus uses workspaces and working units to distribute data among threads which compose a distributed Web application.

Distribution is application driven, allowing applications to specify how to partition data into working units and how to solve conflicts.

Future work on Versus includes a detailed study on performance issues, review of the Versus API and management improvement of extensible meta-data.

References

- [1] Thomas Bergstraesser, Philip A. Bernstein, Shankar Pal, and David Shutt. Versions and Workspaces in Microsoft Repository. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD*

- 1999, *Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 532–533. ACM Press, 1999.
- [2] Fred Douglass, Thomas Ball, Yih-Farn Chen, and Eleftherios Koutsofios. WebGUIDE: Querying and Navigating Changes in Web Repositories. *Computer Networks*, 28:1335–1344, May 1996. In Proceedings of the 5th International World-Wide Web Conference.
 - [3] Fred Douglass, Thomas Ball, Yih-Farn Chen, and Eleftherios Koutsofios. The AT&T Internet Difference Engine: Tracking and viewing changes on the Web. *World Wide Web*, 1(1):27–44, 1998.
 - [4] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. <http://www.globus.org/research/papers/anatomy.pdf>.
 - [5] Allan Heydon and Marc Najork. Mercator: A Scalable, Extensible Web Crawler. *World Wide Web*, 2(4):219–229, 1999.
 - [6] Jun Hirai, Sriram Raghavan, Hector Garcia-Molina, and Andreas Paepcke. WebBase: A repository of web pages. In *Proceedings of the Ninth World-Wide Web Conference*, 1999.
 - [7] The Internet Archive: building an ‘Internet Library’. <http://www.archive.org>.
 - [8] E. James Whitehead Jr. and Yaron Y. Golland. WebDAV: A network protocol for remote collaborative authoring on the Web. In Susanne Bødker, Morten Kyng, and Kjeld Schmidt, editors, *Proceedings of the 6th European Conference on Computer Supported Cooperative Work*, September 1999.
 - [9] Randy H. Katz. Toward a unified framework for version modeling in engineering databases. *ACM Computing Surveys*, 22(4):375–408, 1990.
 - [10] Microsoft SQL Server - SQL Server home. <http://www.microsoft.com/sql/default.asp>.
 - [11] Tamino - The XML power database. <http://www.softwareag.com/tamino/>.
 - [12] TUMBA - Temos Um Motor de Busca Alternativo. <http://www.tumba.pt>.