Next: Introduction

Location-Independent Naming for Virtual Distributed Software Repositories

Shirley Browne, Jack Dongarra, Stan Green, Keith Moore
Theresa Pepin, Tom Rowan, and Reed Wade
University of Tennessee
Eric Grosse
AT& Bell Laboratories

November 11, 1994

Abstract:

A location-independent naming architecture for network resources has been designed to facilitate organization and description of software components accessible through a virtual distributed repository. This naming scheme enables easy and efficient searching and retrieval, and it addresses many of the consistency, authenticity, and integrity issues involved with distributed software repositories by providing mechanisms for grouping resources and for authenticity and integrity checking. This paper details the design of the naming scheme, decribes a prototype implementation of some of the capabilities, and describes how the scheme fits into the development of the National Software Exchange, a virtual software repository that has the goal of providing access to reusable software components for high-performance computing

- Introduction
- Publishing and Name Assignment
- Name Resolution and File Mirroring
- Authenticity, Integrity, and Consistency of Resources
- Prototype Implementation
- Conclusions and Future Work
- Glossary of Terms
- References
- *About this document ...*

Jack Dongarra

Mon Jan 30 10:42:57 EST 1995

Next: Publishing and Name **Up:** Location-Independent Naming for Virtual **Previous:** Location-Independent Naming for Virtual

Introduction

Well-maintained software repositories are central to software reuse because they make high-quality software widely available and easily accessible. One such repository is Netlib, a collection of high-quality publicly available mathematical software. Netlib, in operation since 1985, currently processes over 300,000 requests a day. Netlib is serving as a prototype for development of the National Software Exchange (NSE), which has the goal of encompassing all High Performance Computing Consortium (HPCC) software repositories and of promoting reuse of software components developed by Grand Challenge and other scientific computing researchers.

Many repositories, previously maintained centrally, have evolved into virtual repositories that serve as directories to distributed collections. For example, the GAMS Software Repository, once a central repository [1], is now a virtual repository that catalogs software maintained by other repositories [2]. Another distributed repository, the NASA/GSFC/ESS Software and Information Exchange (accessible at `http://farside.gsfc.nasa.gov/ESS/`), stores some of the catalogued software packages locally, but for packages at remote sites stores just descriptions with pointers. Growth in the popularity of the Internet and the World Wide Web, as well as the wide availability of WWW client and server software, has accelerated the shift from centrally maintained software repositories to virtual, distributed repositories. A provider site (either the maintainer or a mirror site) need only make the file available from an FTP, Gopher, or HTTP server for it to be accessible from a WWW client.

The main advantage of distributing a repository is to allow the software to be maintained by those in the best position to keep it up-to-date. Also, copies of popular software packages may be mirrored by a number of sites to increase availability (e.g., if one site is unreachable, the software may be retrieved from a different site) and to prevent bottlenecks.

A distributed, virtual repository should appear as a centralized, well-organized repository. An effective search interface for a distributed software repository allows a user to search for software without knowing at what site the software is located. A mirrored file should appear once in a list of such results, rather than once for each mirrored copy. Yet a searcher should still enjoy the reliability and performance benefits of mirroring - i.e., be able to try alternative locations to retrieve a search hit or retrieve the closest copy of the file.

Distributed maintenance and mirroring of software introduces challenges as well as benefits. Maintaining the quality of software and of indexing information and presenting a uniform searching and browsing interface become much more difficult. Location-independent naming facilitates the organization, finding, and retrieval of software in a distributed repository, and can be used to provide consistency, authenticity, and integrity guarantees.

The WWW mechanism of specifying a file by its Uniform Resource Locator (URL) poses several difficulties for a virtual software repository. URLs are inadequate for ensuring the consistency and currency of mirrored copies. A URL for an independently mirrored copy of a software package may point to an out-of-date copy and give no indication that it is not up-to-date Consistency between a set of files that are meant to be used together may also become a problem. For example, the Netlib Software Repository provides dependency checking that allows the user to retrieve a top-level routine plus all routines in its dependency tree (i.e., those routines that are called directly or indirectly by the top-level routine). Another example is a graphical parallel programming environment that relies on an underlying parallel communications support package. The problem becomes more complex when different pieces might be retrieved from different physical repositories. Ideally, the user should be able to have a consistent set retrieved automatically without having to scan documentation to verify that compatible pieces have been retrieved.

Distributing the repository also poses challenges for searching. A centrally maintained repository can easily run an indexing and search engine that provides a search interface to the repository contents With the current WWW setup, however, the user has a choice of searching the various distributed repository sites individually or of using a general purpose WWW search engine such as WebCrawler, Lycos, or the World Wide Web Worm.

Most of the above problems can be alleviated by implementing a location-independent naming architecture that includes mechanisms for authenticity and integrity checking. We have designed a naming scheme in which the binding between a name and file contents is unchangeable and verifiable A name may be resolved to multiple, mirrored copies In the case where it represents a set of files, a name may be resolved to a list of other names. The record for a resource that includes the name as well as other descriptive information may be signed by the publisher so that users may verify the authenticity of a retrieved resource This paper describes the design of our naming architecture. We also describe our implementation of a protoype name-to-location service and of a modified WWW client that does name resolution. A glossary of terms used in this paper is included as an appendix.

Next Up Previous

Next: Publishing and Name **Up:** Location-Independent Naming for Virtual **Previous:** Location-Independent Naming for Virtual

Jack Dongarra
Mon Jan 30 10.42 57 EST 1995

Next: Name Resolution and **Up:** Location-Independent Naming for Virtual **Previous:** Introduction

Publishing and Name Assignment

A user who contributes a resource (e.g., a software package or a document) by making it available via our virtual software repository is said to have *published* the resource. Two types of location-independent names may be assigned by the publisher to the resource. For a resource that is made available as a file or a set of files, a Location Independent File Name (LIFN) may be assigned. A LIFN refers to the particular sequence of bytes making up that file, or in the case of a set of files, to the set of LIFNs for those files. Once a LIFN has been assigned to a particular sequence of bytes, that binding may not be changed. Various ways of enforcing this requirement are discussed in Section 4.

The other possible type of name is the Uniform Resource Name (URN), which is a long-lasting name that may be used by humans to refer to some network-accessible resource, be it a Web page or a telnet interface. The exact contents of the resource named by a URN can change. For example, accessing a URN for "the current version of LAPACK" would give a different result following a new release of the LAPACK package.

The LIFN and URN name spaces are subdivided among several *publishers*, also called *naming authorities*, who are responsible for ensuring the uniqueness of names assigned within their portions of the name spaces.

A name is formed by concatenating the registered naming authority identifier and the unique string assigned by the naming authority. For the Netlib naming authority, whose identifier is `netlib`, the unique string for a LIFN consists of the MD5 fingerprint [5] of the file. This fingerprint is a 128-bit quantity resulting from applying the MD5 function to the contents of the file. The function is designed to make it computationally infeasible to find a different sequence of bytes that produces the same fingerprint. The unique string used for a URN is the unique name by which the resource is known in the Netlib Repository - for example, `lapack` for the current version of the LAPACK package. The LIFN and URN are formatted as

```
LIFN:<publisher id>:string
URN:<publisher id>:string
```

The publisher may provide a description for each name it assigns. The description may include information such as title, author, abstract, etc. Suggested attributes for software resources include programming language and system requirements, and for mathematical software, the GAMS classification [1]. A description may also include a `supersedes` field if the file supersedes a previous one (e.g., a new version of a software package). The description for a LIFN should include an MD5 or similar fingerprint, if that fingerprint is not part of the LIFN itself. To enable authentication, the entire description may be cryptographically signed, as discussed in Section 4. Portions of the description may

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.