The Harvest Information Discovery and Access System

C. Mic Bowman, *Member of the Technical Staff, Transarc, Inc.*
Peter B. Danzig, *Assistant Professor, CS Dept., U. Southern California*
Darren R. Hardy, *Professional Research Assist., CS Dept., U. Colorado - Boulder*
Udi Manber, *Professor, CS Dept., U. Arizona*
Michael F. Schwartz, *Associate Professor, CS Dept., U. Colorado - Boulder*

ABSTRACT

It is increasingly difficult to make effective use of Internet information, given the rapid growth in data volume, user base, and data diversity. In this paper we introduce *Harvest*, a system that provides a scalable, customizable architecture for gathering, indexing, caching, replicating, and accessing Internet information.

1. Introduction

Tools like Gopher and WWW make it easy to publish and browse information on the Internet. Making effective use of this information, however, can still be quite hard for the following reasons:

- It's difficult to locate relevant information.
- Popular information leads to network and server bottlenecks.
- Current tools provide little support for structured, complex data.

Harvest addresses these problems through a set of customizable tools for gathering information from diverse repositories, building topic-specific content indexes, flexibly searching the indexes, widely replicating them, and caching objects as they are retrieved across the Internet. The system interoperates with Mosaic and with HTTP, FTP, WAIS, and Gopher information resources.

Harvest permits flexible construction of information services that use both the network and information servers efficiently. It can be configured to automatically collect and summarize related objects from around the Internet into a large collection or to collect, summarize, and hand annotate a tiny, specialized collection. Harvest automates many of the tasks needed to integrate information from disparate repositories and formats. Harvest's architecture makes efficient use of Internet servers and network links, and our measurements indicate that Harvest can reduce server load, network traffic, and index space requirements by one to two orders of magnitude, compared with previous systems.

2. Demonstration Brokers

Harvest index servers are called *Brokers*. Demonstration Brokers are described below. Note that the current Harvest system uses an index/search engine optimized for space efficiency rather than search speed. In the near future we will make indexers with the opposite tradeoff available.

- **Networked Information Discovery and Retrieval** (NIDR) software and documents, and a software + documents index built by cascading the separate indexes into a combined index (at no additional network or server load). These indexes underscore the scaling advantages of topic-specific indexing. For example, the query ‘‘approximate’’ will locate agrep (an approximate match tool embedded in Harvest’s indexing system), while the same query at our more general Computer Science technical reports index (below) locates many unrelated papers.

- **Computer Science technical reports**. This index covers content summaries of almost 21,000 reports, published in a variety of formats (ASCII, PostScript, DVI, HTML, etc.). Content summaries support more powerful searches than the titles/abstracts covered by previously existing CS technical report indexes (such as those offered by Monash University, the University of Indiana, and the University of Karlsruhe). The current index is possible because Harvest provides a more efficient distributed indexing architecture.
- Documents referencing the **Santa Fe Institute** time series competition data. This index demonstrates Harvest’s structured indexing capability and index customizability: in addition to supporting the usual content summary index, the SFI time series broker allows users to search by time series reference. These references were generated by a corpus-specific script attached to the indexing process that matches each document content summary against approximately 70 regular expressions, to heuristically determine the referenced time series.
- **PC Software**. This index demonstrates Harvest’s ability to incorporate information in a variety of formats from other sources, including high quality, manually-generated information sources. Because each indexed site uses a somewhat different format, we used Harvest’s customizable extraction features to collect indexing information in site-specific ways, and place this information into a uniform format. As a result of this effort, we were quickly able to incorporate high quality indexing information about more than 25,000 publically available PC software distributions. This index provides better search support than more general-purpose software indexes (such as Archie), because it contains conceptual descriptions of a focused collection of information. For example, searching for ‘‘batch programming language’’ will locate the "RAP" package, while Archie could only locate this object if you searched for "RAP".

  We have also built Gatherer translation scripts for some other manually created indexing information formats, including the ‘‘Linux Software Map’’ (LSM) format and the the Internet Anonymous FTP Archives IETF Working Group (IAFA) format. At present we have a Broker runing for LSM data but none for IAFA data, because there are not yet enough sites using the IAFA format to warrant building a Broker.
- **World Wide Web Home Pages**. We used Harvest to recognize then index over 7,000 WWW home pages among the several hundred-thousand documents available via the World Wide Web. Because we index content summaries rather than only HTML anchors and links, this index captures much of the content of Web sites without having to collect every last Web document - providing a useful index at lower cost than that incurred by the World Wide Web Worm or Lycos™.

You can also browse and search the list of available Harvest servers (including instances of Gatherers, Brokers, Object Caches, and Replication Managers) by contacting the **Harvest Server Registry**.

3. Technical Overview of System

Most current indexing systems place unnecessary load on information servers and network links because they use expensive object retrieval protocols to gather indexing information, and they do not coordinate information gathering among themselves. Harvest provides a more efficient means of gathering and distributing indexing information, and supports the easy construction of diverse types of indexes. The Figure 1 illustrates the overall Harvest architecture.

Harvest's index gathering efficiency derives from a combination of optimized gathering software and a flexible scheme for sharing gathered information among indexes that need it. A Harvest Gatherer collects indexing information, while a Broker provides an incrementally indexed query interface to the gathered information. Gatherers and Brokers can be arranged in various ways to achieve flexible and efficient use of the network and servers:

- A Gatherer is designed to run at the Provider site, saving a great deal of server load and network traffic.
- A Gatherer can also access an information provider from across the network using the native FTP, Gopher, or HTTP protocols. This arrangement is primarily useful for interoperating with systems that do not run the Harvest software. It experiences the same server and network inefficiencies as discussed above, although the gatherered information can be shared by many Brokers in this arrangement.
- A Broker can collect information from many Gatherers, to build an index of widely distributed information.
- A Gatherer can feed information to many Brokers, saving repeated gathering costs.
- Brokers can retrieve information from other Brokers, in effect cascading indexed *views* from one another, using the Broker's query interface to filter/refine the information from one Broker to the next.

Harvest's flexibility for constructing diverse indexes derives from its customizable subsystems, which allow index builders to control how indexing information is extracted, indexed, and searched.

Gatherers and Brokers communicate using an attribute-value stream protocol called the Summary Object Interchange Format (SOIF), which delineates streams of object summaries.

To provide scalable access support, Harvest replicates indexes and caches retrieved objects. The Replication subsystem can also be used to divide the gathering process among many servers (e.g., letting one server index each U.S. regional network), distributing the partial updates among the replicas.

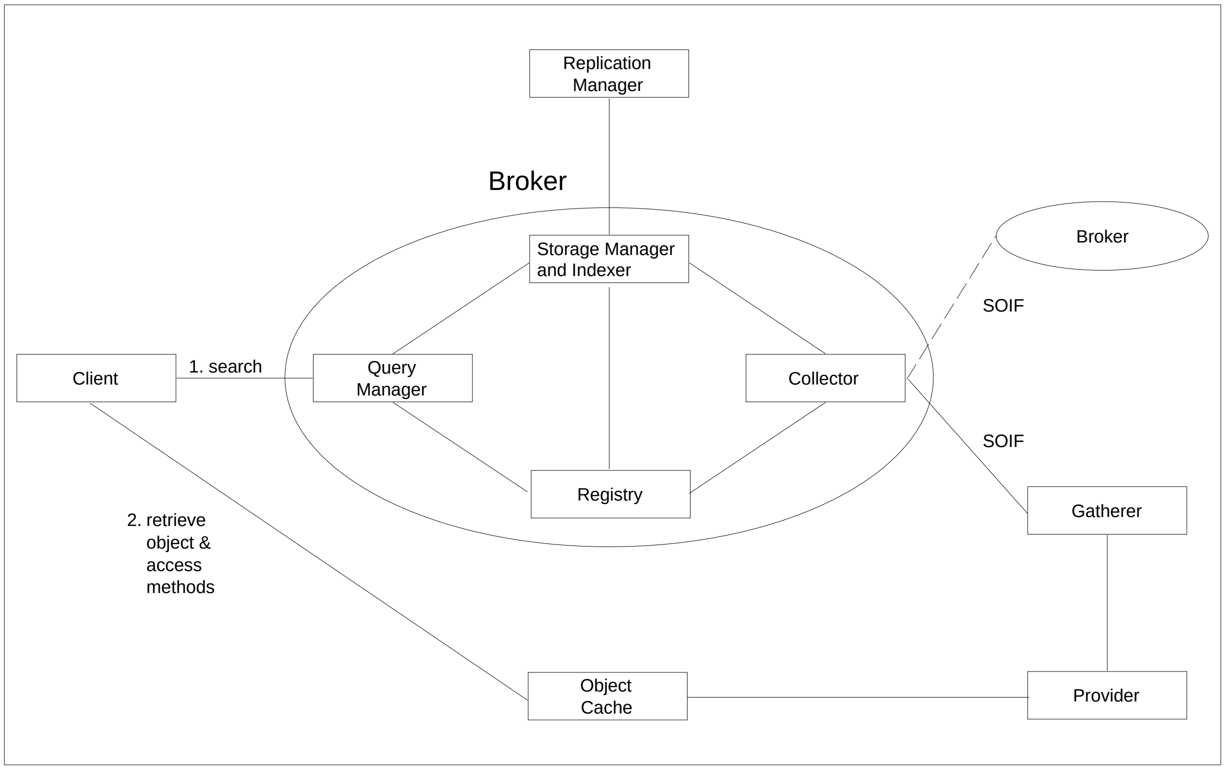

Figure 1 − Overall Harvest architecture

Harvest provides a top level Broker called the Harvest Server Registry (HSR), which registers information about each publicly accessible Harvest Gatherer, Broker, Cache, and Replicator in the Internet. The HSR is useful when constructing new Gatherers and Brokers, to avoid duplication of effort. It can also be used when looking for an appropriate Broker at search time, and to locate Caches and Replicators.

4. Subsystems

Harvest consists of the following subsystems:

- Gatherer
- Broker
- Index/Search Subsystem
- Replicator
- Object Cache

4.1. Gatherer

The Gatherer provides an efficient and customizable way to collect indexing information. We discuss each issue below.

Most current indexing systems generate excessive load on remote sites because they retrieve indexing information via Gopher, HTTP, and FTP which all require heavyweight operations like forking separate processes to retrieve each object. Current indexing systems also cause excess network traffic because they retrieve entire objects yet discard most of the information once it reaches the indexing site (retaining only HTML anchors in an index, for example).

The Harvest Gatherer addresses these inefficiencies in several ways. First, a Provider can run a Harvest Gatherer to export content summaries of its public corpus of information. The Gatherer periodically scans for new or modified resources and adds the content summaries to a local cache. These summary objects can be retrieved from the Gatherer's cache in a single, compressed stream (rather than requiring separate requests for each object). Providing remote access to pre-filtered, incrementally updatable indexing information, and transmitting this information in a single, compressed stream improves the efficiency of network transmissions. These features reduce server and network load by several orders of magnitude each when building indexes.

The Gatherer addresses flexibility through the Essence customizable information extraction system. Essence can unnest presentation layer encodings (such as compressed "tar" files) into constituent files, recognize each file, and extract information in different ways depending on the file types. For example, it can find author and title lines in LaTeX documents, and symbols in object code. More importantly, it allows users to easily customize the type recognition, presentation unnesting, candidate selection, and information extraction phases for use in particular situations.

4.2. Broker

The Broker provides an indexed query interface to gathered information. Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes. The Broker records unique identifiers and time-to-live's for each indexed object, garbage collects old information, and invokes the Index/Search Subsystem when it receives an update or query. In the future, brokers will also provide facilities to identify and remove duplicates and near duplicates.

The Broker can gather objects directly from another Broker using a bulk transfer protocol. It also supports a remote administration interface.

4.3. Index/Search Subsystem

To accommodate diverse indexing and searching needs, Harvest defines a general Broker-Indexer interface that can accommodate a variety of search engines. The principal requirements are that the backend support Boolean combinations of attribute-based queries, and that it support incremental updates. One could therefore use a variety of different backends inside a Broker, such as WAIS or Ingres.

We have developed two Index/Search subsystems for Harvest, each optimized for different uses. Glimpse supports space-efficient indexes and flexible interactive queries, while Nebula supports fast searches and complex standing queries. At present, only Glimpse is used in the Harvest demonstration Brokers and distributed with the Harvest source distribution.

Glimpse uses pointers to adjustable-sized occurrence blocks, achieving very space efficient indexes (typically 3-7% the size of the data being indexed, compared with 100% in the case of

WAIS). It also supports fast and incremental indexing. Moreover, is supports Boolean, regular expression, and even approximate queries (i.e., allowing misspellings).

With Nebula, each object is represented as a set of attribute/value pairs. There is a separate index (and possibly a separate indexing mechanism) per attribute tag. Nebula also supports a view notion involving a standing query against an object database. This allows information to be filtered based on query predicates. It is easy to refine and extend the predicates over time, and observe changes interactively.

4.4. Replicator

Harvest provides a weakly consistent, replicated wide-area file system called mirror-d, on top of which Brokers are replicated. Mirror-d itself is layered atop a hierarchical, flooding update-based group communication subsystem called flood-d.

Each mirror-d instance in a replication group occasionally floods complete state information to its immediate neighbors, to detect updates that flood-d failed to deliver, possibly due to a long-lasting network partion, site failure, or failure of a flood-d process. Mirror-d implements eventual consistency: if all new updates ceased, the replicas eventually converge.

Flood-d logically floods objects from group member to group member along a graph managed by flood-d itself. Each flood-d instance measures the end-to-end network bandwidth achievable between itself and other flood-daemons running at other group member sites. A master site within each group constructs and reliably distributes either a two-connected or a three-connected, low diameter, logical topology of the group members.

A flood-daemon can belong to many groups, making it possible to construct hierarchies of groups and to stitch otherwise independent groups together by sharing two or three common members.

4.5. Object Cache

To meet ever increasing demand on network links and information servers, Harvest includes a hierarchical Object Cache.

The Cache sends "query" datagrams to each neighbor and parent, plus an ICMP echo to an object’s home site, and chooses the fastest responding server from which to retrieve the data. It caches Gopher, HTTP, and FTP objects, plus recent DNS name-to-address maps. It runs as a single, event-driven process. Both I/O to disk and to its clients is non-blocking. The Cache returns data to its clients as soon as the first few bytes of an object fault their way into the Cache. For ease of implementation, the Cache spawns a separate process to retrieve FTP files, but retrieves HTTP and Gopher objects itself. The Cache separately manages replacement of objects on disk and objects loaded in its virtual address space. It also keeps all meta-data for Cached objects in virtual memory, to eliminate access latency to the meta-data. Since the Cache is single threaded but otherwise non-blocking, page faults are the only source of blocking.

The cache uses TTL-based coherence. Clients request objects through the the proxy-HTTP interface.

Our measurements indicate that the cache runs approximately twice as fast as the CERN object cache, and the distribution of object retrieval times has a much shorter "tail" than CERN’s cache.

Our work on the Object Cache subsystem was motivated in part by a simulation study we performed using NSFNET backbone trace data.

5. Software Availability

We are currently soliciting Harvest beta-test sites for the following purposes:

- Running Gatherers, particularly at large archive sites;
- Running topic-specific Brokers; and
- Running Caches, particularly at backbone networks, regional networks, and stub networks (such as a campus or corporate network).

If you are interested in becoming such a beta-test site, please send mail to *harvest-dvl@cs.colorado.edu*.

We will make Version 1.0 of the Harvest source code publically available around the middle of October 1994.

6. Credits and Acknowledgements

Harvest was designed and built by the Internet Research Task Force Research Group on Resource Discovery (IRTF-RD). IRTF-RD consists of Mic Bowman, Peter Danzig, Udi Manber, and Michael Schwartz (IRTF-RD chair). Darren Hardy is a Professional Research Assistant on the project.

Many students have contributed to this project: Rajini Balay, William Camargo, Anawat Chankhunthod, Bhavna Chhabra, Gabe Dalbec, Dante De Lucia, Chanda Dharap, Burra Gopal, James Guyton, Allan Hundhausen, Paul Klark, Shih-Hao Li, Cheng-Che Lue, Dave Merkel, Chuck Neerdaels, John Noble, John Noll, Katia Obraczka, Mark Peterson, and Kurt Worrell.

IRTF-RD is supported primarily by the Advanced Research Projects Agency, with additional support from AFOSR, Hughes, NSF, and Sun. The information contained in this document does not necessarily reflect the position or the policy of the U.S. Government or other sponsors of this research. No official endorsement should be inferred.

7. Why the Name

We call the system *Harvest* to connote its focus on reaping the growing crop of Internet information. While Harvest's initial focus is on resource discovery, the architecture encompasses the more general issue of making effective use of information. In addition to resource discovery this includes:

- view-based customizable information spaces (ala the Nebula Broker subsystem);
- scalable access to popular data (ala the Harvest Object Cache);
- an object-oriented style of access to information objects (ala the Harvest Object Protocol); and
- reducing confusion in the information space, by filtering duplicate information (currently work-in-progress).

8. Bibliography

(1) The current paper was based closely on the Harvest home page (*http://rd.cs.colorado.edu/harvest/*), though the home page will continue to evolve.

(2) The Harvest system is discussed in more depth in the paper:

C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber and Michael F. Schwartz. Harvest: A Scalable, Customizable Discovery and Access System. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, July 1994.

(3) Harvest is based on the architectural motivations described in the paper:

C. Mic Bowman, Peter B. Danzig, Udi Manber and Michael F. Schwartz. Scalable Internet Resource Discovery: Research Problems and Approaches. Communications of the ACM, 37(8), pp. 98-107, August 1994.

(4) The Harvest Gatherer's customized information extraction capability is supported by the Essence system, described in the paper:

Darren R. Hardy and Michael F. Schwartz. Customized Information Extraction as a Basis for Resource Discovery. Technical Report CU-CS-707-94, Department of Computer Science, University of Colorado, Boulder, March 1994.

(5) One of Harvest's Index/Search subsystems is the Glimpse system, described in the paper:

Udi Manber and Sun Wu. GLIMPSE: A Tool to Search Through Entire File Systems. Proceedings of the USENIX Winter Conference, pp. 23-32, San Francisco, California, January 1994.

(6) Another of Harvest's Index/Search subsystems is the Nebula typed file system, described in the paper:

C. Mic Bowman, Chanda Dharap, Mrinal Baruah, Bill Camargo and Sunil Potti. A File System for Information Management. Proceedings of the Conference on Intelligent Information Management Systems, Washington, DC, June 1994.

(7) The Harvest Replication subsystem is described in the paper:

Peter Danzig, Katia Obraczka, Dante DeLucia and Naveed Alam. Massively Replicating Services in Autonomously Managed Wide-Area Internetworks. Technical Report, University of Southern California, January 1994.

(8) Our work on the Object Cache subsystem was motivated in part by a simulation study we performed using NSFNET backbone trace data. This study is described in the paper:

Peter B. Danzig, Richard S. Hall and Michael F. Schwartz. A Case for Caching File Objects Inside Internetworks. Proceedings of the SIGCOMM '93, pp. 239-248, San Francisco, California, September 1993.

(9) The Harvest Object Protocol is described in the paper:

Bhavna Chhabra, Darren R. Hardy, Allan Hundhausen, Dave Merkel, John Noble and Michael F. Schwartz. Integrating Complex Data Access Methods into the Mosaic/WWW Environment. Second International World Wide Web Conference, Chicago, Illinois, October 1994.

9. Author Biographies

C. Mic Bowman is a Member of the Technical Staff at Transarc Corporation. He received his Ph.D in Computer Science from the University of Arizona in 1990. From 1990 to 1994 he was an assistant professor at the Pennsylvania State University. His research interests include descriptive naming systems for local and wide-area file systems, structured file systems, resource discovery, and protocols for remote computing. Bowman can be reached at mic@transarc.com.

Peter B. Danzig is an Assistant Professor of Computer Science at the University of Southern California. He received his Ph.D in Computer Science from the University of California, Berkeley in 1990. His current research addresses on the measurement and performance debugging of Internet services, distributed system architectures for resource discovery, and flow and admission control for packet communication networks. Danzig can be reached at danzig@usc.edu.

Darren R. Hardy is a Professional Research Assistant in the Computer Science Department at the University of Colorado. He received his M.S. in Computer Science from the University of Colorado, Boulder in 1993. He specializes in network resource discovery, distributed systems, and information retrieval. Hardy can be reached at hardy@cs.colorado.edu.

Udi Manber is a Professor of Computer Science at the University of Arizona. He received his Ph.D in Computer Science from the University of Washington in 1982. His research interests include design of algorithms, pattern matching, computer networks, and software tools. Manber can be reached at udi@cs.arizona.edu.

Michael F. Schwartz is an Associate Professor of Computer Science at the University of Colorado. He received his Ph.D in Computer Science from the University of Washington in 1987. His research focuses on international-scale networks and distributed systems. Schwartz chairs the Internet Research Task Force Research Group on Resource Discovery (IRTF-RD), which built the Harvest system. Schwartz can be reached at schwartz@cs.colorado.edu.

Contact author: Mike Schwartz (schwartz@cs.colorado.edu).