Harvest

by Sarah Elizabeth Burcham

As the Internet is inundated with new users, and the amount of data available on the net proliferates, quickly locating sources of information on a specific subject becomes increasingly difficult. As Mark R. Nelson observed in a *Crossroads* article last year [3], ``The tools to support resource identification and use have not increased in effectiveness as rapidly as the quantity of available information has increased." Problems resulting from inefficiently implemented indexing and search methods include network bottlenecks, excessive server load on information provider sites, and large numbers of irrelevant matches to queries.

Two commonly implemented approaches to indexing systems on the Internet are name-only indexing (e.g., archie) and full-content indexing (e.g., WAIS). While name-only indexing allows users to search just the names of the files being indexed, full-content (also referred to as ``full-text") indexing allows the entire contents of the files to be searched. Both approaches have their problems, however. Name-only indexing, though efficient in its use of index space, is limited in its utility by the appropriateness of a document's filename. A significant drawback of full-content indexing is that the number of irrelevant matches to a query increases in proportion to the number and size of the files being indexed. In addition, full content indexes tend to be rather large, thus making it infeasible to create a master index of many sites.

As an alternative to existing indexing and searching approaches, the Internet Engineering Task Force Group on Resource Discovery (IETF-RD) has introduced Harvest, optimized for reducing inefficient use of available system resources. So named ``to connote its focus on reaping the growing crop of Internet information," [1] Harvest is a highly customizable ``set of tools to gather, extract, organize, search, cache, and replicate relevant information across the Internet" [2]. Measurements of performance achieved from the Harvest team's research are astounding in the resulting reduction of strain on network resources. The following statistics are taken from the Harvest FAQ [2]. Harvest reduces network traffic by a factor of 59. Employing Harvest decreases server load by a factor of 4 while obtaining indexing information, but, more impressively, by a factor of 6600 while sending this data to a remote indexer. Furthermore, index space requirements are reduced by a factor of 43.

Existing indexing/searching systems are two-part systems involving a provider (an information site running standard services such as FTP, Gopher, and HTTP) and a remote indexer which performs the tasks of gathering, organizing, and making accessible material from providers. Excessive loading of information providers and the networks linking a remote indexer to providers is a collective result of several inefficiencies of existing indexing systems. Most providers fork a separate process for each object requested. If only a small number of objects are accessed, this separate forking is almost transparent to a provider; however a query requesting hundreds of objects will cause a noticeable drain on a provider's resources. Networks linking existing indexers and a provider are loaded by the retrieval of entire objects which are requested by the indexer. Much of the retrieved information is not pertinent to the query that requested it, and yet network resources are used to transfer the large percentage of unnecessary data. Another inefficiency of existing indexing schemes is the lack of information sharing/ cooperation between indexes. An index will gather information from providers, regardless of whether an index already exists for the requested objects. The combination of these inefficiencies results in network bottlenecks and unnecessary server loading. In order to minimize this loading, the indexer within Harvest is split into two parts, the gatherer and the broker. The gatherer collects information from a provider and then transmits it to a broker via a stream protocol called Summary Object Interchange Format (SOIF). Among other functions, brokers provide an HTML query interface to gathered information received from either another broker or a gatherer.

A broker can build an index of information collected from single or multiple gatherers and/or other brokers. ``This Gatherer/Broker design provides an efficient network pipe facility, allowing information to be gathered, transmitted, and shared by many different types of indexes, with a great deal of flexibility in how information is filtered and interconnected between providers and brokers" [1]. Broker indexing can be updated within a given broker without the need to completely reconstruct the index, thereby saving the duplication of information gathering efforts. This is accomplished by requesting only those objects from a provider that have been altered within a given time period, or, if time stamps are not supported, by comparing cryptographic checksums. A variety of search engines, including some versions of WAIS, may be used within a Harvest broker to optimize indexing for different purposes. Glimpse and Nebula, two search/index subsystems designed for use with Harvest, optimize for small indexes and efficient lookup, respectively. Glimpse, Harvest's default engine, supports boolean queries, approximate searches (allowing partial word or phrase matches in the scope of a search), and case sensitivity. A broker may also contain a replicator, whose job it is to replicate heavily used servers, relieving the load on the server being replicated. An obvious candidate for one such replication is the Harvest Server Registry (HSR) (http://harvest.cs.colorado.edu/brokers/hsr/query.html). The HSR registers and maintains an index of each Harvest instance of gatherers, brokers, caches, and replicators on the Internet. If one is interested in creating a broker or merely interested in exploring available index brokers, the HSR is a good starting place.

As with the broker, the gatherer is designed to reduce loading. This reduction in loading is substantially dependent on the proximity of an information gatherer and provider. If a gatherer must access a provider from across a network, the inefficiencies of existing indexing systems are also applicable to Harvest, with the exception that information sharing is still possible. However, if a gatherer is run on the same machine as the server, the localization of certain indexing tasks causes a significant reduction in network traffic. The most important of these tasks eliminates separate forking of processes at the provider for each requested object. Local gatherer software ``scans the objects [at the provider] periodically and maintains a cache of indexing information," thus allowing ``a Provider's indexing information to be retrieved in a single stream, rather than requiring separate requests for each object. This results in a huge reduction in server load" [1]. Local content summarization, accomplished with the Essence system, sidesteps any necessity to transport entire objects to the remote broker. Content-summary indexing is a middle ground between the extremist approaches of name-only and full-context indexing. Of concern, though, is the relative performance of full-context and content-summary indexes. Comparisons of Harvest's resulting index (using content summarization) to that of WAIS (implementing full-content indexing) reveal that approximately 70% of relevant documents found using WAIS are discovered using Harvest, while only using 3-11% the index space and 70% the time required to build the WAIS index [1].

``The main downside to Harvest is it is more complex" [2] than most widely used distributed indexing systems. Luckily the complexity of Harvest is hidden from the end user. However, system administrators will find Harvest more difficult to install than other systems because it requires extensive customization.

Anyone with a WWW client has the ability to access and use available Harvest search indexes. The Harvest User's Manual, software, and demonstrations are available online from http://harvest.cs.colorado.edu/.

My own experience with Harvest has not yielded a noticeable advantage in response time or ease of operation. This is probably because Harvest claims it is friendly to the net as a whole. The next time you have to wait what seems an inordinate amount of time for a response to what seems a simple query, you might have occasion to wish that more users on the net were employing a utility that was so conservative with the resources we all must share. Bandwidth on the net is expanding at a far slower pace than the increase in traffic. Harvest takes this into account, and in so doing represents what is necessarily going to become the emerging philosophy in the development of distributed utilities.

1.

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, August 26, 1994 Department of Computer Science, University of Colorado - Boulder. ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/HarvestJour.ps.Z

2.

FAQ about Harvest, http://harvest.cs.colorado.edu/harvest/FAQ.html

3.

Mark R. Nelson. We have the information you want, but getting it will cost you! Held Hostage by Information Overload. *Crossroads* September 1994. http://www.acm.org/crossroads/xrds1-1/mnelson.html