Obelix Searches Internet Using Customer Data

Veljko Milutinovic and Predrag Knezevic, University of Belgrade
Bozidar Radunovic, Swiss Federal Institute of Technology
Steve Casselman and John Schewel, Virtual Computer Corp.

One major problem with purchasing through the Web is locating reliable suppliers that offer the exact product or service you need. In the usual approach, you access an indexing-based search engine, specify keywords for the purchase, and initiate the search. The outcome is typically a list ranked according to keyword matches—useful, but not always helpful.

Keyword matches provide only one ingredient to finding the right Web sites. The ranking should also consider the satisfaction of previous customers purchasing from those sites, customer profiles, and customer behavior.

The Obelix search engine uses reconfigurable technology to apply customer satisfaction data obtained from the Internet service provider infrastructure to refine its search criteria. The Obelix system collects data about customer activities, calculates a customer satisfaction index, and updates the search engines with its findings.

THE CONCEPT

Consider the following scenario: An ISP advertises that it has the appropriate infrastructure to collect customer satisfaction data, process it, and make it available to search and ranking engines used by potential customers. Companies offering high-quality products or services readily buy their Internet access from this provider, because taking advantage of the satisfaction of past customers makes sense for such companies.

Incorporating customer satisfaction

The browser implements the customer satisfaction specifiers. Developers modify the browser so that it can collect data about clicks to select a page, cache a page, purchase from a page, and so on. Since many successful browsers (such as Netscape) are now public domain, this is not a problem. Even privately owned companies such as Microsoft will likely permit ISPs to upgrade their proprietary browsers if customers demand these customer satisfaction specifiers. The modified browser collects this data for every Web site hosted at the ISP.

A clock records the time of each purchase so that the related statistical analysis can include time. In addition, the analysis weighs the data. For example, selection gets one point; caching, two points; the first purchase, four points; and so on. If the second purchase follows quickly enough after the first or if the density of purchases increases (both signs of a good product or service and high customer satisfaction), follow-up purchases get eight or more points.

Collecting customer profiles

We can extend this scenario to include customer profiles that provide even more information for a search engine to use in refining its searches. When customers first visit an e-commerce site, they answer certain profile-related questions. A search engine uses this information to affect its ranking of lists. The search engine treats those customers who do not answer the questionnaire as average customers. In other words, customers whose profiles are not available obtain the same ranking of products for the given inquiry. Customers whose profiles are available may obtain different rankings, even though their inquiries are identical.

Tracking customer behavior

Another ingredient we can include is customer behavior, which lets the profile be changeable and traceable in time. By analyzing a customer's past actions, we can extrapolate to predict new actions. This allows better efficiency of the entire product.

Developers can design various algorithms to implement approaches that are more or less sophisticated than the one presented in this scenario. Real-market experiments can show the advantages and drawbacks of these different algorithms (during the development phase) and test performance (during the pilot exploitation phase).

Reconfigurable chips allow user-tailored Internet searches. The first step is collecting customer data. After that, the search engine leads the way.

RECONFIGURABLE TECHNOLOGY

This concept advances the state of the art and depends on several innovations. As we discuss each one, we stress the con-

cept implementation phase, in which reconfigurable chips represent the key technology for efficient implementation.

If we implemented all three algorithmic phases—data collection, creation of weighted sums, and database updating—in software, the ISP would need one PC for the provider function and hundreds to generate real-time customer satisfaction data. Using reconfigurable hardware, on the other hand, is far less expensive. The provider needs only one PC for the provider function and a cabinet with space for tens of reconfigurable hardware boards.

OBELIX

Obelix is an industry-university project that addresses customer satisfaction data. The Obelix search engine uses a slightly modified client-server approach. A modified Web browser informs Obelix servers about client actions through the datagram connection. We chose a connectionless protocol to better handle the high information workload. The loss of a single packet is of minor importance to the search server, and a connectionless protocol reduces communication overhead. Each packet created by the Web browser contains information about the URL and the customer-performed action (saving, printing, mailing, and so on) with its score.

The Obelix engine receives and processes the packets. In addition to a search engine such as Altavista or Lycos, the search server site also includes the Obelix-server-collected information about user actions. Obelix consists of a machine that accepts peripheral component interconnect (PCI) cards (for example, a PC or server), along with several virtual channel connection Hot II boards based on a Xilinx 4000 or Xilinx Spartan, plugged into the PCI slots (*Virtual Computer Corp. HOT II Product Guide*, Virtual Computer Corp., Reseda, Calif., Jan. 2000; http://www.vcc.com/Hotii.html). The number of Hot II boards depends on the system capacity and traffic load, and could vary in a relatively large range (Direct Hit Search Engine, Dec. 1999, http://www.directhit.com/about/products/search_engine.html; and V. Milutinovic, *Infrastructure for E-Business on the Internet*, CRC Press, Boca Raton, Fla., to be published in 2001, http://galeb.etf.bg.ac.yu/~vm/). The information acquisition process involves three phases:

- collecting the packets,
- calculating the sums, and
- transferring the results to the database.

Obelix implements each phase as a separate Hot II configuration.

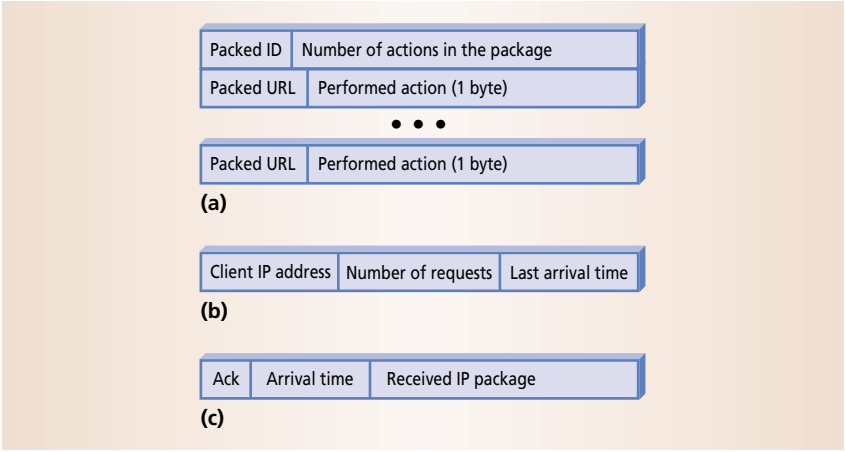

Figure 1. Obelix data structure for (a) information packed in an IP packet during the collection phase, (b) an event table for each Obelix server board, and (c) a packet stored in main memory by a fault-tolerant system.

Collecting information

A modified Web browser packs information into an IP packet of the form shown in Figure 1a. The URLs are packed with static Hoffman compression. Hence, an application never needs to unpack URLs on the server side because the same URLs from different clients have the same code. The actual codes are predefined on a large statistical sample of URLs. User actions cover most user operations with a Web page. Server actions in the current version include visiting a page, saving an entire page, saving part of a page (image or other object), printing, following links from a page, cutting or copying, bookmarking, providing custom dialogue for page ranking, and recording the time spent visiting a page.

A packet passes from the server's network card along the PCI bus toward the appropriate Hot II board. The board receives this package and stores it in memory for later processing.

In designing the collection phase, we also must consider three important issues: load balancing, fake-data detection, and fault tolerance.

If the system has several boards, we need a policy of assigning packets to boards. One possible policy is based on the sender's IP number; another is server-controlled distribution to an idle board. The first solution suffers from loss of packets if a destination board is processing data and thus cannot accept any incoming packets. The second solution minimizes the number of lost packages but generates higher traffic on the PCI bus.

Serving in the real world, the Obelix system is open to different abuses. Web site owners might generate fake traffic to boost their Web site scores. The Obelix server includes detection and prevention of these abuses. Each board includes an event table in its memory, with the structure shown in Figure 1b.

The event table prescribes a certain number of actions that each client can send during a defined period of time; others are ignored. Obelix updates the last arrival time field for each new action. If the difference between the new arrival time and the last arrival time is smaller than the defined period of time, the action is ignored.

Another important issue is the system's fault tolerance. If all Hot II boards are busy calculating sums, they can lose a

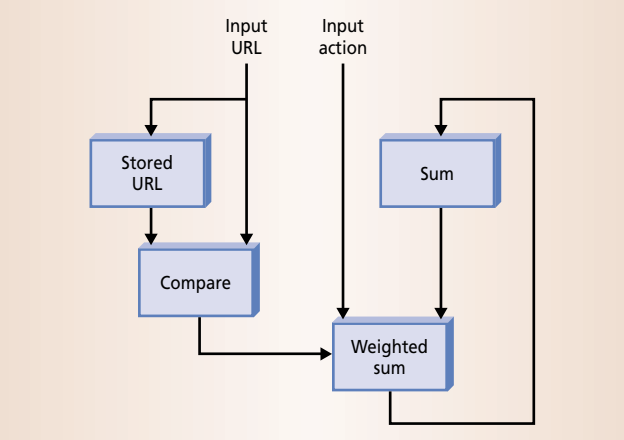

Figure 2. Structure of the basic processing unit. The left side involves finding a URL that has already been stored; the right side shows the calculation of a weighted sum, according to the input action.

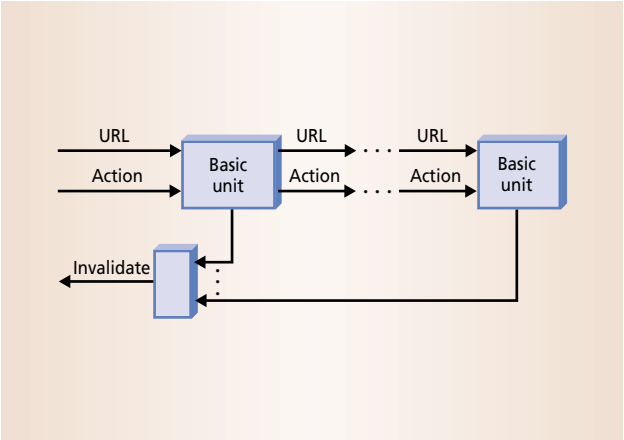

Figure 3. Daisy chain of basic units. You can increase the speedup by connecting more basic units in the chain.

packet arriving through the PCI bus. To correct this problem, Obelix implements two policies: one that permits packet loss and one that doesn't. Systems with high workloads can perhaps tolerate the loss of some portion of user information; such loss is sometimes even preferable to reduce the communication load.

On the other hand, some applications cannot tolerate the loss of data, so we developed a fault-tolerant algorithm. A fault-tolerant system stores packets in main memory (apart from storing them on the boards), using an entry like that shown in Figure 1c.

If a board collects the package from the PCI bus, it sends the package ID to the CPU and sets the Ack bit to one. Package ID is a 16-bit word randomly defined by the client. It is not unique in any sense; rather, it is random enough to reduce possible collision and loss of packages. After some time, the CPU goes through the table and processes all unacknowledged entries, deleting the rest.

Calculating weighted sums

When a board receives a certain number of packages, it starts processing them. This processing calculates a weighted sum from all actions related to a certain URL, where each action has its own weight according to its importance (printing is more important then visiting, for example). Naturally, not all occurrences of a URL appear on the same board, so collecting the weights from various boards requires extra processing in the database system. Despite the extra processing, calculating the partial sums of the weights in Hot II boards greatly reduces the time. The board's Xilinx chip has several basic units, as shown in Figure 2.

These basic units connect in a daisy chain, as shown in Figure 3. A unit compares the stored URL with a URL on the input. If they are the same, the unit updates the weighted sum and disables the input to the Hot II board's memory. Otherwise, it forwards the package to the next unit. If the URL is not stored in the unit, the unit stores the input (the URL and the action), updates the sum, and disables the data in the memory. In this case, the speedup depends on the number of basic units. After finishing the

pass, the basic unit makes a new entry into the memory with the URL and a calculated sum. The board later forwards this record to the main memory and to the database system.

Updating the database

This is the last processing phase. It is implemented as an output function of the Hot II reconfigurable board. The board sends the prepared data (for each URL) to the database. Later on, the search engine will use this data to improve and speed up its search for related documents.

To test the quality of the Obelix system search service, we modified the MS Windows version of the Netscape Navigator Web browser to collect and send information about user actions to the Obelix server. The search server, currently a software-only Web application running on the same Linux server as the Obelix simulator, is based on the Infoseek engine.

We chose Infoseek because it outputs the accuracy of each item it finds. The search server later treats this accuracy as a frequency score. The search server takes input queries from the Web and forwards them to the Infoseek engine. Then the server collects the resulting page and extracts URLs and percentage scores out of it. Afterwards, it calculates the Obelix results.

This work is an example of a synergistic cooperation of hardware and software. Future research should focus on using reconfigurable technology to refine searches on the basis of customer profiles and customer behavior. ✸

Veljko Milutinovic *is a professor of electrical engineering at the University of Belgrade. Contact him at vm@etf.bg.ac.yu.*

Predrag Knezevic *is a PhD student in the School of Electrical Engineering at the University of Belgrade. Contact him at pedjak@opennet.org.*

Bozidar Radunovic *is a PhD student at the Swiss Federal Institute of Technology in Lausanne, Switzerland. Contact him at bokir@europe.com.*

Steve Casselman *is the president and CEO of Virtual Computer Corp., Reseda, Calif. Contact him at sc@vcc.com.*

John Schewel *is the vice president for marketing at Virtual Computer Corp. Contact him at jas@vcc.com.*

Editors: Jerzy W. Rozenblit, University of Arizona, ECE 320E, Tucson, AZ 85721; jr@ece.arizona.edu; and Sanjaya Kumar, Honeywell Technology Center, MS MN65-2200, 3660 Technology Dr., Minneapolis, MN 55418; sanjaya.kumar@honeywell.com