Parallel Architectures for MEDLINE Search

Rajendra V. Boppana
University of Texas at San Antonio, USA

Suresh Chalasani
University of Wisconsin at Parkside, USA

Bob Badgett
University of Texas Health Science Center at San Antonio, USA

Jacqueline A. Pugh
University of Texas Health Science Center at San Antonio, USA

INTRODUCTION

It is often extremely difficult in clinical practice to find answers in a timely manner to specific questions that arise in the management of patient health issues. The central repository of medical research is the MEDLINE database. The time needed to search and *review* results from MEDLINE may range from 20 minutes to 2½ hours (Lucas et al., 2004); this greatly exceeds the two minutes that clinicians typically have available to answer questions during clinical care (Ely et al., 1999). Consequently, clinicians rarely use MEDLINE (Ely et al., 1999); when they use MEDLINE, they do not use it well (Hersh & Hickam, 1998).

In 1997, the National Library of Medicine launched PubMed, which gave the first public and free access to MEDLINE with links to full texts of articles (Free Web-based access to NLM databases, 1997). To reduce the time needed to search and review results, SUMSearch was developed by Badgett, Paukert, and Levy (2001) to automatically examine the results from an initial MEDLINE search and revise queries multiple times to extract more relevant information. Thus, each query from a user may result in up to eight queries by SUMSearch of MEDLINE and other resources. An overview of SUMSearch and other search techniques may be found in the companion article (Badgett, Chalasani, Boppana & Pugh, 2007). However, owing to the restrictions on the access to MEDLINE by the National Library of Medicine, the refined queries need to be issued sequentially, and the rate of queries needs to be throttled. Due to Internet delays accumulating during multiple searches, SUMSearch's response time to a query can be as much as 30 seconds. This delay prevents adding to SUMSearch additional searches that are needed.

In this chapter, we describe a parallel architecture for MEDLINE database integrated with search refinement tools to facilitate accurate and fast responses to search requests by users. The proposed architecture, to be developed by the authors, will use low-cost, high-performance computing clusters consisting of Linux-based personal computers and workstations (i) to provide subsecond response times for individual searches and (ii) to support several concurrent queries from search refinement programs such as SUMSearch.

BACKGROUND

Several commercial companies have implemented fast search engines that run on a cluster of computers and sift through a large volume of content to find relevant information. For example, there are more than 20 billion Web pages registered with the Google search index as of 2006. Google uses sophisticated and proprietary techniques to find and return relevant Internet documents requested by the user. Fast searching can be accomplished by distributing the problem of searching among multiple computers that search simultaneously in various parts of data. Distributed processing and distributed databases have been significant research topics in past years. However, special techniques to optimize the response time of searching are required. For example, loading data to be searched into memory before any searching begins has been explored (Chalasani & Boppana, 2005). Constructing index trees based on keywords to speed up searches

has also been explored by several researchers (Melnik, Raghavan, Yang & Garcia-Molina, 2001; Yu, Cuadrado, Ceglowski & Payne, 2003). Searching in the context of clinical setting was studied by Hersh and Hickam (1994), who showed that simple word searching is an effective means to search for medical literature. Hersh, et al. (2001) showed that Boolean searching and vector space searching are almost equally effective in the context of medical searches.

However, no comprehensive implementation that combines all these various techniques for fast searches has been reported in literature. Most such implementations are commercial and, hence, proprietary in nature. In this chapter, we will combine techniques such as (a) indexing large databases using keywords and content, (b) in-memory loading and searching of databases, (c) distributing content to be searched on multiple computers, and (d) increasing the precision of searches using finite refinements to achieve fast searching of MEDLINE data.

The rest of this chapter is organized as follows. Section 3 describes parallel architectures for the MEDLINE database. Section 4 discusses performance implications for the parallel MEDLINE implementation. Section 5 discusses directions for future research in this area. Section 6 concludes this chapter.

PARALLEL ARCHITECTURES FOR THE MEDLINE DATABASE

Proposed MEDLINE Database Architecture and Implementation

The proposed architecture for a parallel MEDLINE implementation is illustrated in Figure 1. This architecture exploits two types of parallelism:

- **Temporal parallelism.** The incoming requests are distributed equally among the application servers (Appservers). An incoming request is handled completely by a single Appserver.
- **Spatial parallelism.** The MEDLINE database is equally distributed among all database servers (DBservers). In response to an incoming search request, the documents corresponding to that search are retrieved from one or more DBservers.

In this architecture, the application server (Appserver) receives the search request. Each application server has the complete index tree for the MEDLINE database. The incoming requests are evenly distributed across application servers by the Gateway server (Sprayer machine). The application server quickly searches its index tree for the keywords contained in the request. The index tree search identifies the documents relevant for those keywords and the database servers (DBservers) on which those documents reside. The Appserver then requests the DBservers for the identified documents. DBservers return the documents to the Appserver, which then combines the results from various keywords and sends a response to the requestor. This procedure is indicated in Figure 2.

We need to address three critical issues for successful implementation of MEDLINE and to facilitate subsecond search times: (i) implementation of the search engine, (ii) implementation of the database, and (iii) automatic revision of searches. These are discussed further in the next three subsections.

Search Engine Implementation

To facilitate fast and multiple concurrent searches, index lists are often created for a database. An index list is similar to the index at the back of a textbook and indicates the IDs of records that contain a given word. For example, when a request for a list of documents containing the words "hypertension" and "diabetes" is submitted, the index lists for these two words will be examined to identify records that contain both words. Alternatively, a document vector that indicates words (in dictionary order) contained in a document can be used. Given the large vocabulary used in medical publications, the index list approach is preferable for this implementation. The records so identified will be retrieved from the DBservers and composed to generate a response. Figure 3 shows an example index list in the form of a tree search structure. Each keyword (or search term) is a node in this tree. Each node is associated with a list of database servers and the document IDs on that database server that point to relevant documents for that keyword. In the example shown in Figure 3, the relevant documents for the term "diabetes" are documents with IDs 2, 100, and 215 on DBserver 1, and documents with IDs 625, 901, and 1576 on DB server 5. Even though this example indicates a tree search mechanism (to simplify the discussion),