Parallelized Scatter/Gather Browse Search

Greg McLaren*
sybok@abp.lcs.mit.edu
MIT Laboratory for Computer Science

Conventional search techniques applied to large databases have traditionally been narrow attempts to locate documents containing specific *keywords*, or such keyword searches augmented by a *nearest-neighbor search*, which retrieves all documents *near* those containing the desired keywords. Unfortunately, in certain circumstances, the user may not be familiar with the lexicon needed to properly frame a question under a keyword-oriented search paradigm, or may simply want to learn, at some level of detail, what topics are covered in the database. In such a case, where "browsing" might be the best description of the goal, the C*-based[1] *parallelized Scatter/Gather search* for the CM-5 introduced in this paper is ideal.

In a scatter/gather session, the entire database is split, on-the-fly, into a fixed number of *document clusters*, whose member documents are "close" with respect to a pairwise document similarity function. Each cluster is presented to the user through use of a "cluster digest" containing the titles of the documents closest to the center of the cluster, and the most common words in the cluster. Given the digests, the user then can select a number of clusters to focus on, and this subset is gathered together to be used as the new database for the next iteration of the routine.

To scatter a set of documents, a three step process is followed. First, k "seed documents" are found, one for each cluster to be formed. Next, each document is assigned to a seed using the similarity function. Finally, the clusters are refined by computing the cluster centers (not necessarily the same as the seeds) and reclustering by assigning to the nearest center. At each step in the algorithm, the parallel communication overhead is reduced by partitioning the problem correctly.

A key portion of the algorithm is the initial determination of the k "seeds" given n documents. Traditional implementations of seed-based, or partitional, clustering were developed as tools to aid nearest-neighbor search, which required a fine partition. Since $k \to n$ in such circumstances, the algorithms were actually $\approx O(n^2)$ in time. However, by using Scatter/Gather Search as its own paradigm, k can be made small (close to unity), and near-linear time results.

To implement the generation of k seeds with minimal communication overhead and in keeping with the rectangular $O(kn)$ time bound, the problem is decomposed as shown in Figure 1. Computation in the same row occurs in parallel, while vertical data flow (downward) is serially ordered. The corpus of N documents is first split into N/m buckets, each containing m documents potentially on a separate processor. Inside each bucket (and local to a processor), a traditional $O(n^2)$ time *global average agglomerative clustering* routine is run on its m documents, reducing them to m/2 clusters of two documents each. Similarly, in the next stage (depicted by the next row of buckets), m/2 *four-document* clusters are obtained from each bucket of m *two-document* clusters. Since the number of operations in a row is only half as great as in the row above, a brief consid-

*G. McLaren was advised on this matter by Prof. David Karger. G. McLaren's Address: Room 248, 545 Technology Square, Cambridge MA 02139.

[1] C* is a data-parallel version of ANSI C.

eration of the top row shows that the code has time complexity $O(N/m(m^2)) = O(Nm)$. This is rectangular time if $m \to k$, the number of seeds.

Communication is only required in this phase of the code at the highest level, to route agglomerating clusters from one matching site to the next, so it should scale well to a very large number of processors. The subsequent phase, which requires matching each document to the nearest seed, is even more embarrassingly parallel, since each document can be considered isolation from the rest. The scalability of the final phase is noticably lower than the first two, however, since it involves computing *centers* for each of the k (≈ 10) clusters, and in such a calculation each cluster is best placed on its own processor.[2]

When run on a 27,654 document corpus from Grolier's encyclopedia with **k**, the bucket size and final number of clusters, set to 8, the initial implementation takes 224.97 seconds on 64-node partition of the CM-5, 148.77 of that being node (PN) time. The effect of various values of k on the running time is shown in Figure 2.

References

[1] Douglass Cutting, and David Karger, et al. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. 15th ann Int'l SIGIR'92/Denmark, June 1992.

[2] Cutting, Karger, and Pederson. Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections. 16th ann Int'l SIGIR'93/Pittsburgh, June 1993.

[2]To scale past 10 or so processors, the clusters might first be quartered or eighthed, their childrens' centers computed, and the original centers found by vectorially adding the childrens' centers.

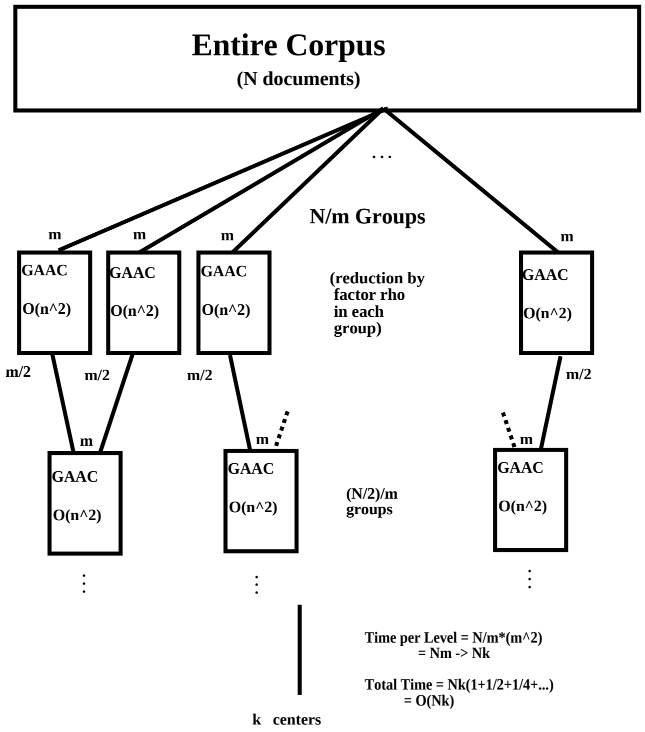

Figure 1: Parallelism and data flow in determination of the K centers (Fractionation).

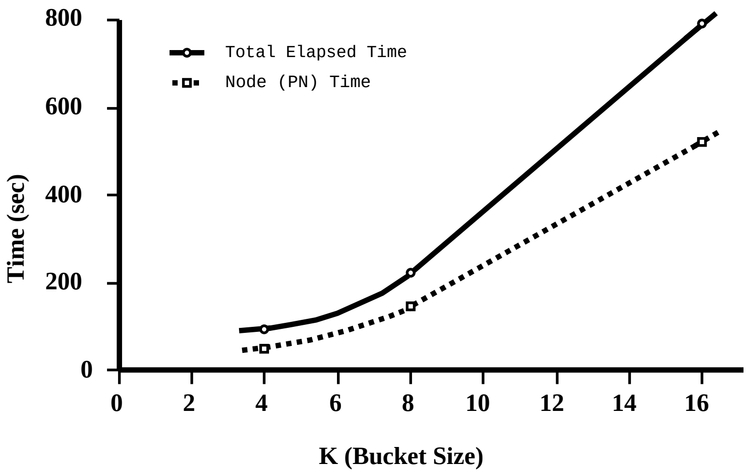

Figure 2: Total and Node (PN) Time for Parallelized Scatter/Gather running on a 64-node partition of the CM-5.