GEØTREE
A Benchmark Test For A General File Management System

by

Douglas A. Kellogg*

Technical Report 73002 February, 1973

University Computer Center
University of Minnesota
Minneapolis, Minnesota
55455

*presently with the Hybrid Computer Laboratory,
University of Minnesota.

This report describes the results of a benchmark test for the file management system FMS described in A General File Management System for the CDC 6600.*

METHOD

The benchmark program, GEOTREE, sets up a file representing two Minnesota counties, Aitkin and Itasca, and then retrieves data concerning land use and water orientation for geographical subsections of the two counties from the file. GEOTREE can be thought of as consisting of four parts: initialization, setting up the file structure, computing summary information at each level of the file, and accessing the file to retrieve this information. Each of these parts requires a separate pass through the file.

Initialization

Initialization is fairly straight forward. A file of 22,000 pages is set up (of which 21,451 are used) with a hash table size of 8280 pages. An overlay is called from GEOTREE which writes blank numbered pages on the disk.

Setting Up File

The main part of the program is concerned with setting up the basic file structure. Data is read in from a tape supplied by the Center for Urban and Regional Affairs (CURA). This data is decoded into parcel

---

* University Computer Center Technical Report UCC004.

names. One problem which occurred was that the tape contains many errors, such as duplicate names, out-of-order names, and illegal names. These problems had to be accounted for one-by-one. The current GEOTREE program allows for all errors that were found, and prints out all duplicate cards. The tree structure of the file consists of the state (at the top) with the two counties as children. Each county has, below it, townships, sections (thirty-six per township), quarter-sections, and parcels (usually four per quarter-section) at the lowest level. The program reads through 80,320 parcel cards, giving a total of over 105,000 entities. Each parcel has a land-use value ranging from one to nine, and a water-orientation ranging from zero to nine. A specially developed hashing algorithm is used so that a township and all its associated entities are stored together in the hash table. This portion of the program has to check whether or not a parcel is present and if not, add it along with all necessary higher entities. Then all linking, e.g. parcel to quarter-section, is done in order to build up the tree.

Summarizing:

The summarizing is done by a single subroutine. The number of parcels in each land-use or water-orientation category are stored in each entity above the parcel level in the tree. Thus, the number of parcels with land-use one, for instance, in a given quarter=section, section, township and county are stored in that quarter-section, section, township and county. This process involves looking at each parcel in the file and adding data to every entity above the parcel level. Since disk addresses are always gotten from pointers and the hash table is never accessed, this part of GEOTREE is much faster than setting up the tree initially.

Retrieval

The retrieval portion of the program is performed by another overlay, which acts just like a separate program. To avoid reading in cards, loops are used to perform one retrieval from each of 148 townships or anything belonging to that township, most of which exist. In order to do a retrieval, the hash table first has to be accessed to determine if the entity desired is in the file, and if it is the entity itself is fetched and the required datum is printed out. Possible retrievals include number of parcels with a given land use or water orientation at a certain level, and the total number of parcels below a certain level. For example, there are 412 parcels with land use one in township 43222 of Aitkin county, and sixteen parcels in the 35th section of the same township. In most situations a user would desire many retrievals from a single township. Because of the use of ECS and in-core slots the additional time necessary to get more information from the same township, or even adjacent townships, after the first retrieval would be relatively small, while no more disk accesses would be needed. So that while the sample retrieval is not the worst possible case, which would be retrievals completely at random instead of in the order the entities were added, it is far from being the best possible case.

RESULTS

The followingtable lists the timings for the four parts of the benchmark program. The CP times are printed out by the program as it runs, and can vary from run to run by a few percent. The next column gives the number of times an ECS buffer is filled or emptied (read/write). These figures are taken from the eighteenth and nineteenth words of

labeled common, respectively, and give an indication of how efficient the program is in terms of PP use, since these values are proportional to the number of disk I/O operations needed. Note that these values depend on the number and size of the ECS slots, as mentioned in the technical report for FMS. In GEOTREE five slots of 300 pages each are used. These are reasonable, but not necessarily optimum, values. Next we have the number of entities accessed, and in the last column there is the average CP time per entity accessed. The different parts of GEOTREE are examples of different types of file accesses: the second part is entity addition, with a GETLOC/ADHASH/STPNTR for each entity; the third part is entity replacement with GETHSH/RPHASH; and the fourth is entity retrieval, with the processes used being GETLOC/GETHSH.

Part	CP Time	reads/writes	no. of entities	average time
1	.50s	0/79	---	---
2	564	105/94	105,000	5.36 ms
3	52	47/47	105,000	.05
4	3	91/0	148	21.9

CORE REQUIREMENT

GEOTREE and all its associated overlays require 62000B(= 25,600) to load and 55700B(- 23,488) to run. An ECS field length of 277000B is used.