Multivariate Geographic Clustering on the World’s First Zero Price/Performance Parallel Computer

Forrest M. Hoffman
Oak Ridge National Laboratory*
P.O. Box 2008
Oak Ridge, TN 37831-6036
423-576-7680 voice
423-576-8543 fax
forrest@esd.ornl.gov

Andrew J. Schultz
University of Tulsa
600 South College Avenue
Tulsa, OK 74104-3189
918-631-2000
schlitz@hpserv.utulsa.edu

William W. Hargrove
Oak Ridge National Laboratory*
P.O. Box 2008
Oak Ridge, TN 37831-6274
423-241-2748 voice
423-241-3870 fax
hnw@esd.ornl.gov

Abstract

The authors present an application of multivariate non-hierarchical statistical clustering to geographic environmental data from the 48 conterminous United States in order to produce maps of regions of ecological similarity, called ecoregions. These maps represent more realistic and finer scale regionalizations than those generated by the traditional technique: an expert with a marker pen. Nine input variables thought to affect the growth of vegetation are clustered at a resolution of one square kilometer. These data represent over 7.7 million map cells in a 9-dimensional data space. Denied the funding for the construction of a Beowulf-style cluster of new PCs on which to perform this analysis, the authors built a 126-node cluster out of surplus PCs--primarily Intel 486 CPUs with a host of different motherboards and connected via 10 Mb/s ethernet--obtained at no cost from federal facilities in Oak Ridge, Tennessee. The authors describe the construction of this unique and heterogeneous cluster. Running RedHat Linux with the GNU compilers and both PVM and MPI, this cluster, aptly named the Stone SouperComputer, is the first parallel computer with a price/performance ratio of zero. After developing a serial version of the iterative statistical clustering algorithm, the authors developed a parallel version of the algorithm which uses the MPI message passing routines. The parallel algorithm uses a classical master/slave organization, performs dynamic load balancing for reasonable performance on heterogeneous clusters, and saves intermediate results for easy restarting in case of hardware failure. In addition to being run on the Stone SouperComputer, the parallel algorithm was tested on other parallel platforms without code modification. Finally, the results of the geographic clustering are presented.

Clustering

Statistical clustering is the division, arrangement, or classification of a number of (non-identical) objects into subgroups or categories based on their similarity. Hierarchical clustering provides a series of
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
divisions, based on some measure of similarity, into all possible numbers of groups, from one single group which contains all objects to as many groups as there are objects, with each object being in a group by itself. Hierarchical clustering, which results in a complete similarity tree, is computer-intensive, and the assemblage to be classified is limited to relatively few objects. Non-hierarchical clustering provides only a single, user-specified level of division into groups; however, it can be used to classify a much larger number of objects.

Multivariate geographic clustering employs non-hierarchical clustering on the individual pixels in a digital map from a Geographic Information System for the purpose of classifying the cells into types or categories. The classification of satellite imagery into land cover or vegetation classes using spectral characteristics of each cell from multiple images taken at different wavelengths is a common example of multivariate geographic clustering. Rarely, however, is non-hierarchical clustering performed on map cell characteristics aside from spectral reflectance values.

Maps showing the suitability or characterization of regions are used for many purposes, including identifying appropriate ranges for particular plant and animal species, identifying suitable crops for an area or identifying a suitable area for a given crop, and identifying Plant Hardiness Zones for gardeners. In addition, ecologists have long used the concept of the ecoregion, an area within which there are similar ecological conditions, as a tool for understanding large geographic areas (Bailey, 1983, 1994, 1995, 1996; Omernick, 1986). Such regionalization maps, however, are usually prepared by individual experts in a rather subjective way, and are essentially objectifications of expert opinion.

Our goal was to make repeatable the process of map regionalization based not on spectral cell characteristics, but on characteristics identified as important to the growth of woody vegetation. By using non-hierarchical multivariate geographic clustering we intended to produce several maps of ecoregions across the entire nation at a resolution of one kilometer per cell. At this resolution, the 48 conterminous United States contains over 7.7 million map cells. Nine characteristics from three categories—elevation, edaphic (or soil) factors, and climatic factors—were identified as important. The edaphic factors are 1) plant-available water capacity, 2) soil organic matter, 3) total Kjeldahl soil nitrogen, and 4) depth to seasonally-high water table. The climatic factors are 1) mean precipitation during the growing season, 2) mean solar insolation during the growing season, 3) degree-day heat sum during the growing season, and 4) degree-day cold sum during the non-growing season. The growing season is defined by the frost-free period between mean day of first and last frost each year. A map for each of these characteristics was generated from best-available data at a 1 km resolution for input into the clustering process (Hargrove and Luxmoore, 1998). Given the size of this input data and the significant amount of computer time typically required to perform statistical clustering, we decided a parallel computer was needed for this task.

**The Stone Souper Computer**

Because of the geographic clustering application and other computational research opportunities, a proposal for internal funding was developed which would support the construction of a Beowulf-style cluster of new PCs, the first such system at Oak Ridge National Laboratory, and the development of the necessary software algorithms. With the proposal rejected and significant effort already expended, we chose to build a cluster anyway using the resources that were readily available: surplus Intel 486 PCs destined for salvage.

Commandeering a nearly-abandoned computer room and scavenging as many surplus machines as
possible—from Oak Ridge National Laboratory, the Y-12 production plant, and the former K-25 site (all federal facilities in Oak Ridge)—we setup a "chop shop" to process machines and proceeded to construct the first no-cost parallel computer system. Aptly named the Stone SouperComputer, after the age-old children's fable entitled Stone Soup (Brown, 1987), the heterogeneous cluster grew slowly to 126 nodes as PCs became available and were either cannabilized or fashioned into acceptable nodes. The nodes contain a host of different motherboards, processors (of varying speed and design, including a handful of Pentiums), controllers, and disk drives. Each has 20 MB of memory, at least 400 MB of disk space (for booting and local file access), and is connected to a private 10 Mb/s ethernet network for inter-cluster communications. In addition, one of the nodes is also connected to the external network for logins and file transfers. Running RedHat Linux, the GNU compilers, and the PVM and MPI message passing libraries for parallel software development, the Stone SouperComputer is the first parallel computer with a price/performance ratio of zero (Hoffman, et. al.).

Figure 1: The Stone SouperComputer: the world’s first zero price/performance parallel computer system.

The system, which is used for parallel program development and running models, is constantly changing. As new versions of a popular non-Linux operating system for PCs are released, better hardware becomes available for assimilation into the cluster as users upgrade their desktop PCs. Staying just behind the curve means the Stone SouperComputer will have a free supply of upgrades indefinitely.

While not offering the performance of most Beowulf clusters, this machine is an excellent platform for developing parallel models which will port directly to other systems and for solving nearly perfectly parallelizable problems like non-hierarchical multivariate geographic clustering.

The Algorithm

In our implementation of non-hierarchical clustering, the characteristic values of the 9 input variables are used as coordinates to locate each of the 7.7 million map cells in a 9-dimensional environmental data space. The map cells can be thought of as constellations of "unit-mass stars" fixed within this 9-dimensional volume. The density of "unit-mass stars" varies throughout the data space. "Stars" which are close to each other in data space have similar values of the nine input variables, and might, as a result, be included in the same map ecoregion or "galaxy." The clustering task is to determine, in an iterative fashion, which "stars" belong together in a "galaxy," the number of which is specified by the user. The coordinates of a series of "galaxy" centroids, or its "centers of gravity," are calculated after each iteration, allowing the "centers of gravity" to "walk" to the most densely populated parts of the data space.
The non-hierarchical algorithm, which is nearly perfectly parallelizable, consists of two parts: initial centroid determination, called seed finding, and iterative clustering until convergence is reached. The algorithm begins with a series of "seed" centroid locations in data space—one for each cluster desired by the user. In the iterative part of the algorithm, each map cell is assigned to the cluster whose centroid is closest, by simple Euclidean distance, to the cell. After all map cells are assigned to a centroid, new centroid positions are calculated for each cluster using the mean values for each coordinate of all map cells in that cluster. The iterative classification procedure is repeated, each time using newly recalculated mean centroids, until the number of map cells which change cluster assignments within a single iteration is smaller than a convergence threshold. Once the threshold is met, the final cluster assignments are saved.

Seed centroid locations are ordinarily established using a set of rules which sequentially examines the map cells and attempts to preserve a subset of them which are as widely-separated in data space as possible. This inherently serial process is difficult to parallelize; if the data set is divided equally among \( N \) nodes, and each node finds the best seeds among it's portion of the cells, and then a single node finds the "best-of-the-best," this set of seeds is not so widely dispersed as a single serially-obtained seed set. On the other hand, the seed-finding process is quite slow on a single node, while the iterations are relatively fast in parallel. It may be foolish, in terms of the time to final solution, to spend excessive serial time polishing high-quality initial seeds, since the centroids can "walk" relatively quickly to their
ultimate locations in parallel. Thus, we opted to "get close" to good initial seeds by serially considering only every hundredth map cell, and then jumping to the relatively rapid parallel iteration process.

The iterative portion of the algorithm is implemented in parallel using the MPI message passing routines (specifically, MPICH from Argonne National Laboratory) by dividing the total number of map cells into parcels or aliquots, such that the number of aliquots is larger than the number of nodes. We employ a classical master/slave relationship among nodes and perform dynamic load balancing because of the heterogeneous nature of the Stone SouperComputer on which the algorithm is run. This dynamic load balancing is achieved by having a single master node act as a "card dealer" by first distributing the centroid coordinates, and then an aliquot of map cells to all nodes. Each slave node assigns each of its map cells to a particular centroid, then reports the results back to the master. If there are additional aliquots of map cells to be processed, the master will send a new aliquot to this slave node for assignment. In this way, faster and less-busy nodes are effectively utilized to perform the majority of the processing. If the load on the nodes changes during a run, the distribution of the work load will automatically be shifted away from busy or slow nodes onto idle or fast nodes. At the end of each iteration, the master node computes the new mean centroid positions from all assignments, and distributes the new centroid locations to all nodes, along with the first new aliquot of map cells. Because all nodes must be coordinated and in-step at the beginning of each new iteration, the algorithm is inherently self-synchronizing.

If the number of aliquots is too low (i.e., the aliquot size is too large), the majority of nodes may have to wait for the slowest minority of nodes to complete the assignment of a single aliquot. On the other hand, it may be advantageous to exclude particularly slow nodes so that the number of aliquots, and therefore the amount of inter-node communication, is also reduced often resulting in shorter run times. Large aliquots work best for a parallel machine with few and/or homogeneous nodes or very slow inter-node communication, while small aliquots result in better performance on machines with many heterogeneous nodes and fast communication. Aliquot size is a tunable parameter, which makes the code portable to various architectures, and can be optimized by monitoring the waiting time of the master node in this algorithm.

Comparisons of algorithm runs on a number of parallel platforms will be presented.

In order to provide some fault-tolerance, the master node saves centroid coordinates to disk at the end of each iteration. If one or more nodes fails or the algorithm crashes for some reason, the program can simply be restarted using the last-saved centroid coordinates as initial seeds, and processing will resume in the iteration in which the failure occurred.

The Results

The clustering algorithm was used to generate maps with 1000, 2000, and 3000 ecoregions. These maps evidently capture the ecological relationships among the nine input variables. This multivariate geographic clustering can be used as a way to spatially extend the results of ecological simulation models by reducing the number of runs needed to obtain output over large areas. Simulation models can be run on each relatively homogeneous cluster rather than on each individual cell. The clustered map can be populated with simulated results cluster by cluster, like a paint-by-number picture. This cluster fill-in simulation technique will be used by the Integrated Modeling Project to assess the health and productivity of southeastern forests.
While our construction of a Beowulf-style cluster was hindered by a complete lack of funding, building one from surplus equipment has proved to be well worth the effort. We have used it to develop and run the clustering algorithm and are making it available to others who are interested in developing and running parallel scientific models. In addition, small universities and colleges, as well as high schools, have contacted us and expressed interest in building clusters from existing or "obsolete" machines. It is clear such "throw-away" equipment represents a hidden resource in schools and businesses. It is possible to make use of this resource because of the grass root efforts behind Linux, the Free Software Foundation, and academia. This kind of Beowulf-style cluster is the ultimate price/performance winner.

Finally, we have demonstrated that while special considerations must be made for algorithms running on relatively-large heterogeneous systems, like dynamic load balancing and fault tolerance, they are not difficult to implement and will result in enhanced performance not just on a single parallel machine, but can be tuned for many different architectures.

References


*Oak Ridge National Laboratory, managed by Lockheed Martin Energy Research Corp. for the U.S. Department of Energy under contract number DE-AC05-96OR22464.

"The submitted manuscript has been authored by a contractor of the U.S. Government under contract No. DE-AC05-96OR22464. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes."