Parallel computer processing in systematics.

Wheeler, W., Janies, D., and De Laet, J.

Pp. 247-250 in
McGraw-Hill yearbook of science and technology 2004.
McGraw-Hill, ISBN 0-07-142784-8.

Parallel computer processing in systematics

A central goal of evolutionary biology is the reconstruction of the tree of life. All life, both living and extinct forms, is thought to have originated from a single ancestor that lived billions of years ago. It is the task of systematics to reconstruct this history, and comparative deoxyribonucleic acid (DNA) analysis is an important tool in this process. When systematists create family trees (or phylogenies) of living things, they draw on all varieties of natural variation such as anatomy, physiology behavior, and molecular biology Each of these types of data provides information on the origin of species, and systematists endeavor to create scenarios of evolutionary history that explain the incredible diversity of life in all these areas. *See* TREE OF LIFE.

Technological advances in DNA sequencing have allowed for the creation of huge comparative databases of genomic data, and these molecular data sets have enabled far-reaching systematic analyses addressing the most fundamental questions about the origin of biological diversity However, evolutionary tree questions are extraordinarily difficult computational problems, and the analysis of DNA sequences is especially complex. As a result, systematists have turned to parallel-processing computers to explore their molecular data. Such high-powered computers can perform simultaneous calculations many hundreds or thousands of times faster than traditional, single central processing unit (CPU) machines. A particular architecture for parallel processing—clustering—is becoming a popular model for high-performance parallel computing in systematics. This model can be cost-effective and efficient, but provides challenges of its own as phylogenetic problems become larger and molecular data sets more complete.

DNA sequence alignment. Alignment is the first step in the process of determining homologies (similarities based on descent from a common ancestor) between sequences of DNA in different organisms. It starts with strings of DNA nucleotides as observed in nature and finishes with a matrix of nucleotides and "gaps" (place-holders for insertions or deletions of DNA). Alignments specify correspondences between DNA bases in different organisms. For example, in the top alignment and top tree of **Fig. 1*a*,** the first G of sequence I corresponds to the first G of sequence III, but not to the first G of sequences II and IV. In the alternative alignment and tree in

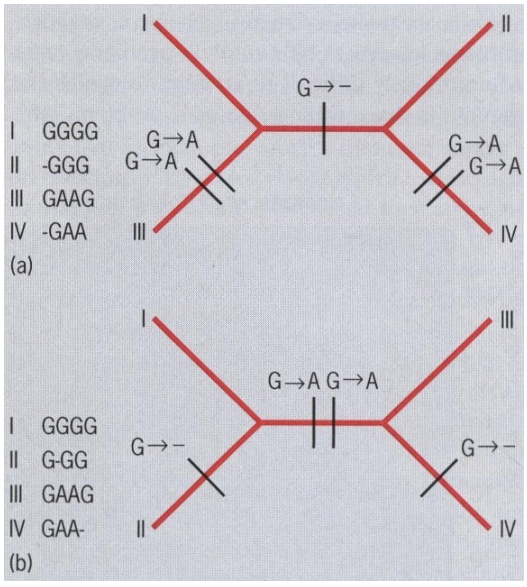

Fig. 1. Alternative alignments for four sequences on a tree. The sequences (GGGG, GGG, GAAG, GAA) are aligned with two evolutionary trees. *(a)* The simplest arrangement for the top alignment is the top evolutionary tree (costing six evolutionary steps, denoted by lines and arrows, because in both scenarios insertion and deletion of nucleotides Is weighted as two nucleotide substitutions) and *(b)* that for the bottom alignment the bottom tree (also costing six evolutionary steps). Mismatching the alignments and trees (top alignment with bottom tree, and vice versa) results in scenarios that cost an additional evolutionary step.

Fig. 1*b,* the same first G of sequence I does correspond to the first G in sequences II and IV (as well as III).

This process of alignment is required for sequence data, as opposed to, for example, anatomical information, for two reasons: (1) DNA sequence data exhibit only four states (adenine, A; cytosine, C; guanine, G; and thiamine, T); and (2) these occur in simple linear sequences. The combination of these two factors can make it difficult to determine which "A" in the sequence from one creature corresponds to that in another. This is usually thought not to be a problem in anatomical data because of the complexity of the structures that are involved. For example, a hand on one creature is sufficiently complex (many states) that it is unlikely to be confused with another feature, such as the vertebral column, on another.

Sequence alignment creates the homology statements that are required to interpret evolution of sequences on evolutionary trees. Since there are so many possible combinations of sequence change even on a single tree, computer algorithms have been devised to create alignments that seek to determine the best scheme of nucleotide homologies on a given tree and, out of all possible trees of relationships, those trees that best explain observed sequences. One measure of best is simplicity, or parsimony. When using this measure, the best alignment (and tree) is that which is simplest in that it implies the least amount of evolutionary change. The alignment and the evolutionary tree are tightly coupled, and in fact cannot be separated; each evolutionary tree implies a potentially unique alignment (Fig. 1).

Parallel processing. DNA sequence alignment and evolutionary tree construction fall into a category of problems known as NP-complete problems, which are notoriously difficult to analyze. (A well-known example is the traveling salesman problem, which involves minimizing the distance traveled in a tour of a number of cities.) Except in the simplest cases, exact solutions cannot be determined, and approximate, or heuristic, solutions are sought. These heuristic solutions themselves can be very time consuming to calculate; hence scientists have turned to parallel processing to analyze their data.

Parallel processing consists of breaking up large amounts of work into smaller tasks and distributing these smaller tasks to different processors (or computers) so that the multiple subproblems are solved simultaneously (as opposed to performing each subproblem sequentially, one on a single processor in turn). With appropriate algorithms, a parallel computer can perform multiple operations simultaneously on separate CPUs. Speedup of calculation time with a parallel computer depends on (1) coupling–. the extent to which algorithms depend on intermediate results that are required to move on to further operations, and (2) overhead—the amount of extra calculations that have to be performed to turn a sequential algorithm into a parallel algorithm. Thus, the efficiency of a parallel algorithm is influenced by coupling and by overhead (**Fig. 2**). Algorithms that have speedups that are close to linear as the number of computers used (cluster size) grows are said to scale well.

Commonly used heuristics in multiple alignment and tree search include the consideration of many candidate trees with randomization—Monte Carlo techniques. Efficient parallel speedup can be achieved by allocating one processor to each of these random trials. This strategy allows an investigator to examine replicates on inexpensive computing clusters of personal computers linked with low-bandwidth, high-latency networks. In this setup the master node allocates the replicates to the slave nodes and assembles they overall result as slave nodes

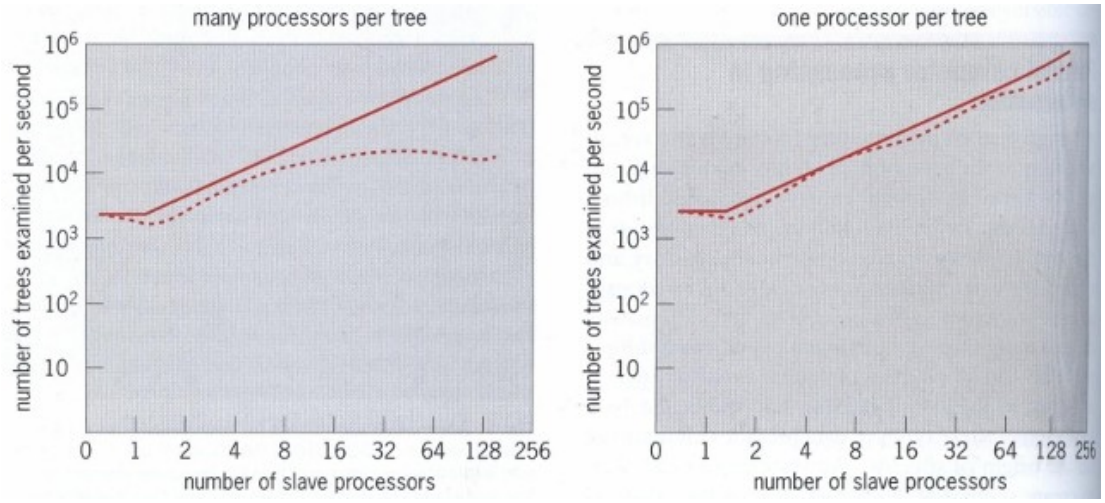

Fig. 2. Graphs illustrating the parallel efficiency of the core algorithms of POY in a large PC/LINUX computing cluster in service at the American Museum of Natural History. The solid plots indicate the best speedup of tree-based alignment (measured in trees examined per second) that can be expected as a function of the addition of processors. The dashed plots indicate the actual speedup during testing. *(a)* Parallel building (using many processors per tree) is inefficient, whereas *(b)* multibuilding (using one processor per tree) is very efficient.

return their partial results. This strategy has been successful in large clusters by reducing overall search time in proportion to available CPUs. However, inefficiency can arise when not all replicates finish at the same time; thus under some strategies many processors might sit idle waiting for instructions while other replicates finish. Moreover, once replicates are finished and the master processor collects results, the best candidate trees are subject to further refinement that relies on intense communication between processors, further limiting parallel efficiency. *See* BAYESIAN INFERENCE (PHYLOGENY).

Cluster computing. As processor and network speeds have increased and costs decreased, a model of parallel computing has arisen which seeks to take advantage of off-the-shelf computers and networks. This model is often referred to as cluster or Beowulf computing. Unlike traditional special-purpose parallel computers, clusters rely on inexpensive hardware and often on open-source software. They rely on numbers—hundreds to thousands of processors, at relatively low-cost—to achieve spectacular computational speed. As low-cost network systems, however, clusters present problems infrequently encountered by "big-iron" parallel machines. This has to do with the variable availability of computers, either due to hardware or software failure or due to scheduling of resources. The latter is most common in screen-saver approaches, in which the appearance of the screen-saver in a networked computer serves as a signal that the computer is available for use in a cluster.

Consider a computer system built of commodity parts that experiences, on average, one failure every 10 years (MTBF, mean time between failures). This makes it a very reliable system for most stand-alone uses. When building a cluster with this same type of computer, however, the outlook changes dramatically. Assume a duster of 365 such nodes. On this cluster, the average time between node failure is just 10 days. In addition, the cluster can also suffer failures in the hardware and the software that connect the nodes, so failure somewhere in the cluster can be expected even more frequently, perhaps on average once a week. This is well below the time that advanced phylogenetic analyses of DNA sequence data would require for completion on that cluster. Hence it is crucial that programs for parallel analysis of DNA sequences be fault-tolerant. Fault tolerance in this context refers to the concept that a cluster program must be able to complete the calculations that are requested even if some parts of the cluster become unavailable while the program is running. This means, in the first place, that the program must be able to detect failures in the duster. In addition, if failures are detected, the program must be able to recover from these automatically: it has to check if it was waiting for partial results from nodes that have become unavailable and, if so, redistribute the work that these nodes were performing to parts of the cluster that are still functioning as expected (**Fig. 3**). Without such provisions, most phylogenetic analyses that would take, a week on the example cluster of the previous paragraph would simply never run to completion.

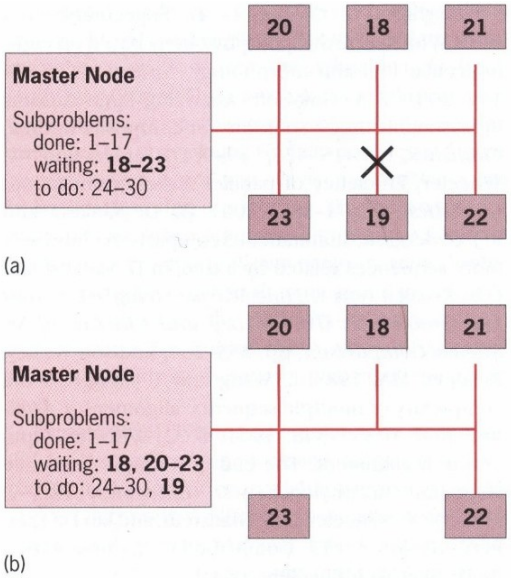

Fig. 3. Example of fault tolerance in cluster computing. ***(a)*** **A cluster that is logically structured, in which one master node and six slave nodes have to solve a problem that can be divided into 30 subproblems that can each be solved on a slave node. When the first 17 subproblems have been solved, the slave node that deals with subproblem 19 becomes unavailable because of a failure in that node or in the connection of that node to the cluster.** ***(b)*** **The master node reacts to this by removing that node from the resources that it will use and by rescheduling subproblem 19.**

Parallel programs can also dynamically check if new nodes become available during a run, and immediately start using these additional resources. Such functionality highly increases the flexibility of the system (as in screen-saver approaches mentioned above). As an example, when a first user starts a job on the cluster, all nodes would typically be assigned to that job. If a second user would then arrive with a second job, the cluster administrator has two options: (1) Wait until the first job is finished and then start the second job. (2) Pull half of the nodes from the first job and assign them to the second job that is started immediately. In the second case, depending on which job is finished first, the other job will dynamically get the nodes that then have become available again. Although the second scenario requires extra calculations (has overhead) to dynamically restructure the nodes that jobs can use, the overall time of completion of the two jobs may be much better because of scaling issues (that is, the counter-intuitive behavior of parallel systems as the number of processors is increased).

For background information *see* CONCURRENT ROCESSING; DEOXYRIBONUCLEIC ACID (DNA); MACROEVOLUTION; MULTIPROCESSING; ORGANIC EVOLUTION; PHYLOGENY; PROTEINS, EVOLUTION OF; TAXONOMY in the McGraw-Hill Encyclopedia of Science & Technology.

Ward Wheeler; Daniel Janies; Jan De Laet

Bibliography. G. Giribet, G. D. Edgecombe, and *W.* C. Wheeler, Arthropod phylogeny based on eight molecular loci and morphology; *Nature,* 413:157-161, 2001; P. A. Goloboff, Analyzing large datasets in reasonable times: Solutions for composite optima, *Cladistics,* 15:415-428, 1999; D. Janies and W Wheeler, Efficiency of parallel direct optimization, *Cladistics,* 17:S71-S82, 2001; D. D. Sankoff and R. J. Cedergren, Simultaneous comparison of three or more sequences related by a tree, in D. Sankoff and J. B. Kruskal (eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison,* pp. 253-264, Addison-Wesley, Reading, MA, 1983; L. Wang and T. Jiang, On the complexity of multiple sequence alignment, J. *Cornput. Biol.,* 1:337-348, 1994; W. C. Wheeler, Optimization alignment: The end of multiple sequence alignment in phylogenetics?, *Cladistics,* 12:1-9, 1996; W. C. Wheeler, D. S. Gladstein, and Jan De Laet, POY, Version 3.0.11, Commandline documentation by De Laet and Wheeler, 2003.