Nucleic Acids Research, 2007, Vol. 35, Web Server issue ***W33–W37***
doi:10.1093/nar/gkm359

MultiPhyl: a high-throughput phylogenomics webserver using distributed computing

Thomas M. Keane[1,2,*], Thomas J. Naughton[3] and James O. McInerney[2]

[1]Pathogen Sequencing Unit, Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Hinxton, CB10 1SA Hinxton, UK, [2]Department of Biology, National University of Ireland, Maynooth, Co. Kildare, Ireland and [3]Department of Computer Science, National University of Ireland, Maynooth, Co. Kildare, Ireland

Received January 8, 2007; Revised April 18, 2007; Accepted April 25, 2007

ABSTRACT

With the number of fully sequenced genomes increasing steadily, there is greater interest in performing large-scale phylogenomic analyses from large numbers of individual gene families. Maximum likelihood (ML) has been shown repeatedly to be one of the most accurate methods for phylogenetic construction. Recently, there have been a number of algorithmic improvements in maximum-likelihood-based tree search methods. However, it can still take a long time to analyse the evolutionary history of many gene families using a single computer. Distributed computing refers to a method of combining the computing power of multiple computers in order to perform some larger overall calculation. In this article, we present the first high-throughput implementation of a distributed phylogenetics platform, MultiPhyl, capable of using the idle computational resources of many heterogeneous non-dedicated machines to form a phylogenetics supercomputer. MultiPhyl allows a user to upload hundreds or thousands of amino acid or nucleotide alignments simultaneously and perform computationally intensive tasks such as model selection, tree searching and bootstrapping of each of the alignments using many desktop machines. The program implements a set of 88 amino acid models and 56 nucleotide maximum likelihood models and a variety of statistical methods for choosing between alternative models. A MultiPhyl webserver is available for public use at: http://www.cs.nuim.ie/distributed/multiphyl.php.

INTRODUCTION

One of the greatest goals of biology is the reconstruction of the tree of life. Recent advances in high-throughput sequencing technologies have seen an exponential increase in the amount of publicly available genomic data. The huge increase in genomic data requires us to develop efficient high-throughput methods for analysing this data. Nowadays it is usual for whole-genome phylogenetic studies to comprise of thousands of different genes and search through large numbers of possible phylogenies. For example, in supertree analysis, researchers first construct a large number of fully optimized phylogenies with partially overlapping taxa and combine these input trees to produce more comprehensive phylogenetic hypotheses (e.g. 1,2). One of the most accurate techniques for constructing phylogenetic trees is using maximum likelihood (ML) estimation (3).

Recently, there have been significant advances in model-based tree search methods that can be mainly attributed to algorithmic improvements rather than hardware improvements. Programs such as Phyml (4), RaxML (5), MrBayes (6) and IQPNNI (7) have each introduced better and faster heuristics that have been shown to repeatedly outperform traditional ML programs such as fastDNAml (8) in terms of computing time and topological accuracy. The impressive results presented by Guindon and Gascuel (4) serve as an example of the extent to which these algorithmic improvements have made it possible to use ML programs to construct relatively large phylogenies quickly using standard hardware. Despite these improvements, it can still take a considerable amount of time to perform large-scale phylogenetic analyses involving thousands of alignments using only a single processor. One popular method of addressing this limitation is to take a number of closely coupled processors to form a dedicated processing cluster for performing high-throughput phylogenetic analysis (9–11). Although extremely effective, these clusters can be prohibitively expensive as they often require dedicated processors and specialized personnel to maintain these systems. On the other hand, it has been shown in a recent study by IBM that Windows and UNIX servers are idle for approximately 95 and 85% of the day respectively (12) while another study showed that on average, desktop machines are idle for between 60 and 80% of the day (13).

*To whom correspondence should be addressed. Tel: +44 (0) 1223 494954; Fax: +44 (0)1223 494919; Email: tkeane@cs.nuim.ie

The term distributed computing is often taken to mean computing using a non-dedicated pre-existing pool of heterogeneous processors that are not localized in a single room or building and can fail arbitrarily. By harnessing the spare clock cycles of multiple semi-idle desktop machines (a semi-idle machine is a machine primarily used by another individual that, because of the nature of the usage, is idle some of the time, e.g. a computer in a university teaching laboratory), it is possible to emulate the computing power offered by specialized dedicated hardware at a fraction of the cost (14). Recently, a number of authors have identified the suitability of distributed computing for phylogenetic analysis and shown how it is possible to harness the idle clock cycles of hundreds of standard desktop machines to perform phylogenetic analysis (15,16).

In this article, we introduce MultiPhyl, which is a fully cross-platform high-throughput ML-based phylogeny analysis program that allows researchers to create a virtual phylogenetic supercomputer from a group of semi-idle desktop machines. The MultiPhyl webserver is aimed at researchers who wish to perform phylogenetic analysis of large numbers of alignments in short amounts of time but do not have access to dedicated high-performance computational resources.

MULTIPHYL

MultiPhyl is one of a number of applications that runs on our general-purpose distributed computing system (16,17). The overall design of the system is based on the client-server model. This model describes a system consisting of a single server computer and a number of client computers. Our system is divided into three separate pieces of software: server, client and remote interface (18). Apart from the server, the system is completely dynamic meaning that new donor machines (a donor machine is owned by someone that offers their machines spare clock cycles to MultiPhyl) can be added at any time and existing donor machines can fail arbitrarily with no adverse effect to the computations. We have deployed the distributed system across our university campus using the idle computing resources of several hundred desktop computers (see http://www.cs.nuim.ie/distributed/statistics.php for details).

The overall flow of control in MultiPhyl can be broken up into three distinct stages: model selection, tree searching and bootstrapping. The user uploads a group of n nucleotide or amino acid alignments in either FASTA or PHYLIP file format. MultiPhyl makes extensive use of the Java-based Phylogenetic Analysis Library (PAL) v1.5 (19) to perform all likelihood calculations. Each of the analysis stages (model selection, tree searching and bootstrapping) operates completely asynchronously across different alignments, thus achieving maximum speedup and avoiding performance bottlenecks. One of the essential requirements for ML phylogenetic inference is a model of amino acid or nucleotide substitution. It has been shown that using incorrect or suboptimal substitution models can produce less accurate or incorrect phylogenies when the wrong model of evolution is assumed (20). Therefore, it may be useful to first compute the most suitable model before performing a treesearch. However, model selection is one area that has been overlooked as an essential part of the phylogeny construction process by many other parallel phylogenetic construction programs. We have recognized the importance of integrating model selection into all phylogenetic studies and therefore MultiPhyl allows the user to perform model selection on every alignment prior to all tree searches (20). The task of testing multiple substitution models against an alignment is very amenable to parallelization as each model can be optimized completely independently and the corresponding likelihoods can be collected and analysed at the server to determine the best model. In order to gain the maximum speedup for the model selection stage of the program, we implemented an adaptive parallel strategy where the server first recruits a single donor machine to construct the initial NJ tree to be used to optimize each model. The parallel granularity of a model selection stage is estimated dynamically by recording how many models are optimized by the first donor machine in m min (where the default value of m is 15 min). Although it may seem that the optimal parallel granularity would be to just issue a single model per donor machine, this could have a number of adverse effects on the distributed system such as overloading the non-dedicated network or overloading the server with many simultaneous connections. After the server receives the first set of o optimized models, it issues groups of o models to multiple donor machines to be optimized. As MultiPhyl is based on a network of heterogeneous processors with varying computational abilities, each donor machine has the option to return a partial set of p optimized models (where p is the number of models optimized by the particular machine within 15 min and $p < o$) with the unoptimized models being redistributed to other machines, thereby avoiding a bottleneck that might be caused by a few slow donor machines in the system.

In order to address the trade-off between running time and accuracy when performing large-scale analysis, we have implemented two different tree search algorithms in MultiPhyl and given the user the option to select which algorithm they wish to use to analyse their dataset. The first tree- building algorithm initially constructs an NJ tree and then iteratively improves the trees by repeatedly performing all possible simultaneous NNI and local branch length optimization swaps that improve the likelihood until convergence (4). The model parameters are also optimized after each round of NNI swaps. Finally, all of the branch lengths are globally optimized to improve the final likelihood of the tree. The second tree search algorithm implemented in MultiPhyl allows the user to define a maximal depth of subtree pruning and regrafting (SPR) rearrangements to perform from each node of the tree. The SPR tree search algorithm begins by creating a starting tree by performing the NNI search algorithm outlined above and then iteratively visits each node of the tree and performs SPR rearrangements spanning increasing levels of the tree (beginning at 2 levels and continuing up to a user-defined maximum). In order to accommodate both long tree searches and the varying computational abilities of the available donor machines,

if an individual tree search has not converged after 60 min on a particular machine, then the current state of the tree search is saved and returned to the server. This tree search is then re-issued to another donor machine to continue optimizing the tree. This also ensures that if a donor machine fails (e.g. crashes or is switched off), it is only possible to lose a maximum of approximately 60 min computing time for any tree search.

One significant challenge in a distributed computing environment is the huge disparity of available memory on the donor machines. This is a particular issue in phylogenetics as the amount of memory consumed by ML-based phylogeny programs increases linearly with sequence length and number of taxa. MultiPhyl implements a dynamic memory scheme where the minimum memory requirements of each task is first set to some initial value (default is 60 Mb). When a donor machine requests a work unit, the donor machine can only be issued a particular task (model optimization or tree search) that has a memory requirement less than the amount of memory available to the Java Virtual Machine (JVM) on the donor machine. If a donor machine reports that the JVM ran out of memory while attempting to perform a particular task, the server dynamically increases the minimum memory requirements of that individual task to a level that is suitable for the task. If no machine with sufficient memory can be detected after q attempts (default value for q is 5), then the problem is removed from the system with an appropriate entry in the log file returned to the user. This dynamic scheme ensures that each phylogenetic computation can gain access to the maximum amount of computational resources available to MultiPhyl as the user is not required to decide a priori what types of machines are capable of processing their dataset.

WEBSERVER

The MultiPhyl webserver (http://www.cs.nuim.ie/distributed/multiphyl.php) allows any researcher to upload multiple amino acid or nucleotide alignments (in FASTA or PHYLIP format) to be analysed on our network of computers. A user can upload either a single alignment file or a zip file containing up to 1000 alignments. The user must provide a valid email address to which the results are sent when the analysis is finished. Each alignment is first checked for formatting errors and an email is sent to notify the user if any errors are found. The webserver is set to perform model selection by default and the user chooses which tree search algorithm to use and whether or not to perform bootstrapping. For each alignment, a comprehensive report file is produced with information on the alignment (number of sites, gap content, constant sites), search parameters, model selection ranking information, sequence-composition chi-squared test, final likelihood, human-readable tree, phylip format tree and a bipartition table constructed from the bootstrap replicates.

A user is presented with a web submission form when they log onto the MultiPhyl homepage. To avail of the service, the user first selects their alignment file (or a zip file containing their alignments). The user then uses the check-boxes to specify which tree search algorithm (NNI or SPR) to use for their analysis. Next, the user can select whether to perform bootstrapping on their dataset and enter how many bootstrap replicates to carry out in the text box. All users must enter their name, institution and email address in the text-boxes provided. Finally, the user presses the execute button to start the analysis on the distributed system. Any errors found in the submission information are displayed on the screen immediately.

VALIDATION

In order to benchmark the tree search algorithms implemented in MultiPhyl, we ran MultiPhyl on a single processor and compared it to a number of other prominent tree search programs, using a number of previously published well-known real alignments. We obtained the three datasets used to benchmark parallel fastDNAml (11) corresponding to 50 (1858 bp), 101 (1858 bp) and 150 (1269 bp) taxa. We also used the 218RDPII (4128 bp) dataset used to benchmark Phyml (4) and the 150ARB (3188 bp), 193V (465 bp), 200ARB (3270 bp) and 250ARB (3638 bp) datasets used to benchmark RAxML (5). The performance of six different phylogeny construction programs across three different architectures was recorded. As MultiPhyl optimizes the base frequencies of the model, Phyml was executed using the 'F' option to also optimize the base frequencies. The final likelihood values and running times of each program using the GTR (21) nucleotide substitution model are given in Table 1. In all cases, the final likelihood values of the trees were calculated using Phyml using the branch and model optimisation option to allow for a fair comparison of likelihood values.

In terms of final likelihood values, the performance of MultiPhyl (NNI) and Phyml is quite similar. Phyml achieves higher final likelihood values in four out of the eight datasets (150SC, 193V, 218RDPII and 250ARB) with MultiPhyl (NNI) achieving higher likelihoods in the other four datasets (50SC, 101SC, 150ARB and 200ARB). As both programs implement very similar tree search algorithms, the differences in performance can be attributed to slight implementation differences of the search algorithm in the two programs. For the smaller datasets (50SC, 101SC, 150SC, 150ARB and 193V), the runtimes of the two programs are quite comparable with MultiPhyl (NNI) producing runtimes up to 1.5 times slower than Phyml. However, the runtimes from the three largest datasets (200ARB, 218RDPII, 250ARB) show more significant differences between the two programs. The performance of the more extensive tree search programs [MultiPhyl (SPR), RAxML and IQPNNI] shows a number of patterns. RAxML clearly outperforms the two other programs in terms of final likelihood values. In five out of the eight datasets (101SC, 150SC, 193V, 200ARB and 218RDPII), RAxML produced the highest likelihoods of any of the programs tested. MultiPhyl (SPR) produced the highest likelihood in one dataset

Table 1. Comparison of final log likelihood values and runtimes (seconds in brackets) of MultiPhyl v1.0.4, RAxML-VI, Phyml v2.4.4, DPRml and IQPNNI v3.0 for a number of previously published datasets

Dataset	MultiPhyl (NNI)	Phyml v2.4.4	DPRml	MultiPhyl (SPR)	IQPNNI v3.0	RAxML-VI
50SC	−43664 (78)	−43691 (66)	−43735 (5994)	−43651 (6777)	−43630 (443)	−43630 (69)
101SC	−73807 (163)	−73818 (145)	−73754 (82935)	−73790 (13674)	−73648 (1703)	−73610 (602)
150SC	−44178 (162)	−44138 (113)	−44082 (141211)	−44172 (13931)	−44061 (2513)	−44024 (342)
150ARB	−76472 (482)	−76489 (323)	−76571 (309434)	−76472 (17441)	−76473 (4117)	−76473 (814)
193V	−64799 (235)	−64532 (263)	n/a	−64802 (13411)	−64413 (2511)	−64407 (905)
200ARB	−103741 (1003)	−103789 (395)	n/a	−103740 (17147)	−103784 (6470)	−103696 (1487)
218RDPII	−155967 (1049)	−155876 (535)	n/a	−155934 (17233)	−155607 (11508)	−155603 (2937)
250ARB	−130530 (1262)	−130488 (619)	n/a	−130518 (17570)	−130271 (15226)	−130287 (3575)

SPR rearrangements were limited to a maximum of 5 branches of the tree in MulitPhyl (SPR). All tests were carried out on an AMD Opteron 2.4 GHz with 1 GB RAM. Phyml was used to calculate the final likelihood values produced by each program. DPRml was not tested on some of the larger datasets due to its excessive runtimes.

(150ARB) with IQPNNI also producing the highest likelihood for the 250ARB dataset. The scale of the algorithmic improvements in recent years is apparent by examining the performance of DPRml compared to all of the other programs. DPRml is based on a traditional tree search algorithm that has been in use for many years (8,16). However, in three of the four datasets that DPRml was executed on, it produced the lowest (worst) likelihood values of all of the programs.

CONCLUSION

As the number of fully completed genomes increases, we are becoming greatly interested in performing whole-genome phylogenetic analyses by examining the evolutionary history of large numbers of gene families. Despite the recent improvements in ML-based tree search programs, it can still take a long time to perform a full ML phylogenetic analysis of multiple genes on a single computer. In order to address these computational challenges, we have recently developed the first high-throughput implementation of a distributed phylogenetics platform, MultiPhyl, capable of using the idle computational resources of many heterogeneous non-dedicated machines to form a phylogenetics supercomputer. MultiPhyl allows users to upload multiple amino acid or nucleotide alignments and perform the tasks of ML model selection, tree searching, and bootstrapping. We have shown that the tree search algorithms implemented in MultiPhyl are quite comparable to the fastest serial phylogenetic programs. In future versions of MultiPhyl, we intend to integrate some of the other computationally intensive processes that are performed when carrying out a typical phylogenomic analysis such as multiple sequence alignment and simultaneous estimation of alignment and tree parameters. We also intend to incorporate more complex models of sequence evolution such as heterogeneous mixture models of nucleotide and amino acid evolution. MultiPhyl is currently installed on several hundred desktop computers in our university campus and we encourage other institutions to also install MultiPhyl. Perhaps the true significance of the MultiPhyl webserver is that it allows researchers who otherwise do not have access to high performance computational hardware (e.g. in developing countries) the opportunity to perform large phylogenomic studies.

ACKNOWLEDGEMENTS

Financial support for this project was provided by the Irish research council for Science, Engineering and Technology. We would like to thank all of the computer technicians at the Department of Computer Science and the Computer Center at NUIM for their continued support and assistance in deploying the software across the university campus. Funding to pay the Open Access publication charges for this article was provided by the Department of Biology, National University of Ireland, Maynooth, Co. Kildare, Ireland.

Conflict of interest statement. None declared.

REFERENCES

1. Philip,G.K., Creevey,C.J and McInerney,J.O. (2005) The opisthokonta and the ecdysozoa may not be clades: stronger support for the grouping of plant and animal than for animal and fungi and stronger support for the coelomata than ecdysozoa. *Mol. Biol. Evol.*, **22**, 1175–1184.
2. Fitzpatrick,D.A., Logue,M.E., Stajich,J.E. and Butler,G. (2006) A fungal phylogeny based on 42 complete genomes

derived from supertree and combined gene analysis. *BMC Evol. Biol.*, **6**, 99.
3. Delsuc,F., Brinkmann,H. and Philippe,H. (2005) Phylogenomics and the reconstruction of the tree of life. *Nat. Rev. Genet.*, **6**, 361–375.
4. Guindon,S. and Gascuel,O. (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst. Biol.*, **52**, 696–704.
5. Stamatakis,A.P., Ludwig,T. and Meier,H. (2005) RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, **21**, 456–463.
6. Ronquist,F. and Huelsenbeck,J.P. (2003) MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, **19**, 1572–1574.
7. Vinh,L.S. and von Haeseler,A. (2004) IQPNNI: Moving fast through tree space and stopping in time. *Mol. Biol. Evol.*, **21**, 1565–1571.
8. Olsen,G. J., Matsuda,H., Hagstrom,R. and Overbeek,R. (1994) fastDNAmL: a tool for construction of phylogenetic tees of DNA sequences using maximum likelihood. *Comput. Appl. Biosci.*, **10**, 41–48.
9. Brauer,M.J., Holder,M.T., Dries,L.A., Zwickl,D.J., Lewis,P.O. and Hillis,D.M. (2002) Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. *Mol. Biol. Evol.*, **19**, 1717–1726.
10. Schmidt,H.A., Strimmer,K., Vingron,M. and von Haeseler,A. (2002) TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, **18**, 502–504.
11. Stewart, C.A., Hart, D., Berry, D.K., Olsen, G.J., Wernert, E.A. and Fischer, W. (2001) Parallel implementation and performance of fastDNAml - a program for maximum likelihood phylogenetic inference. In *Proceedings of SC2001*, Denver, CO, 20–31.
12. Heap, D.G. (2003) Taurus - A Taxonomy of Actual Utilization of Real UNIX and Windows Servers. IBM eServer Library on ibm.com, *Technical White Paper GM12-0191*.
13. Acharya, A., Edjlali, G. and Saltz, J. (1997) The utility of exploiting idle workstations for parallel computing. In *Proceedings of the 1997 ACM SIGMETRICS International Conference on Measurement & Modeling of Computer Systems*, pp. 225–236.
14. Bohannon,J. (2005) Grassroots supercomputing. *Science*, **308**, 810.
15. Stamatakis, A., Ott, M., Ludwig, T. and Meier, H. (2005) DRAxML@home: A Distributed Program for Computation of Large Phylogenetic Trees. In *Future Generation Computer Systems*, **21**, 725–730.
16. Keane,T.M., Naughton,T.J., Travers,S.A.A., McInerney,J.O. and McCormack,G.P. (2005) DPRml: distributed phylogeny reconstruction by maximum likelihood. *Bioinformatics*, **21**, 969–974.
17. Keane,T.M. and Naughton,T.J. (2005) DSEARCH: sensitive database searching using distributed computing. *Bioinformatics*, **21**, 1705–1706.
18. Keane,T.M., Page,A.J., Naughton,T.J., Travers,S.A.A and McInerney,J.O. (2006) Building large phylogenetic trees on coarse-grained parallel machines. *Algorithmica*, **45**, 285–300.
19. Drummond,A. and Strimmer,K. (2001) PAL: An object-oriented programming library for molecular evolution and phylogenetics. *Bioinformatics*, **17**, 662–663.
20. Keane,T.M., Creevey,C.J., Pentony,M.M., Naughton,T.J. and McInerney,J.O. (2006) Assessment of methods for amino acid 60 matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified. *BMC Evol. Biol.*, **6**, 29.
21. Rodriguez,F., Oliver,J.L., Marin,A. and Medina,J.R. (1990) The general stochastic model of nucleotide substitution. *J. Theor. Biol.*, **142**, 485–501.