Chapter 23

High-Performance Computing in Bayesian Phylogenetics and Phylodynamics Using BEAGLE

Guy Baele, Daniel L. Ayres, Andrew Rambaut, Marc A. Suchard, and Philippe Lemey

Abstract

In this chapter, we focus on the computational challenges associated with statistical phylogenomics and how use of the broad-platform evolutionary analysis general likelihood evaluator (BEAGLE), a high-performance library for likelihood computation, can help to substantially reduce computation time in phylogenomic and phylodynamic analyses. We discuss computational improvements brought about by the BEAGLE library on a variety of state-of-the-art multicore hardware, and for a range of commonly used evolutionary models. For data sets of varying dimensions, we specifically focus on comparing performance in the Bayesian evolutionary analysis by sampling trees (BEAST) software between multicore central processing units (CPUs) and a wide range of graphics processing cards (GPUs). We put special emphasis on computational benchmarks from the field of phylodynamics, which combines the challenges of phylogenomics with those of modelling trait data associated with the observed sequence data. In conclusion, we show that for increasingly large molecular sequence data sets, GPUs can offer tremendous computational advancements through the use of the BEAGLE library, which is available for software packages for both Bayesian inference and maximum-likelihood frameworks.

Key words Adaptive Markov chain Monte Carlo, Multipartite data, Generalized linear model, High-performance computing, BEAGLE, BEAST, Pathogen phylodynamics, Data integration, Bayesian phylogenetics, Phylogenomics

1 Introduction

Phylogenomics, a term coined by Eisen and Fraser [13], explores the intersection of evolutionary studies and genomic analyses. Accurate phylogenetic reconstruction using genomic data has important repercussions for answering particular questions in genome analysis, as phylogenomic analyses often involve estimating the underlying evolutionary history of sequences either as an intermediate goal or as an end point. The availability of more and more complete genomes can help to correct for phylogenetic reconstruction artifacts and contradictory results that often appeared in

Maria Anisimova (ed.), *Evolutionary Genomics: Statistical and Computational Methods*, Methods in Molecular Biology, vol. 1910, https://doi.org/10.1007/978-1-4939-9074-0_23,

molecular phylogenies based on a single or few orthologous genes [21]. Expanding the number of characters that can be used in phylogenetic reconstruction from a few thousand to tens of thousands, these large quantities of data lead to reduced estimation errors associated with site sampling, to very high power in the rejection of simple evolutionary hypotheses and to high confidence in estimated phylogenetic patterns [4].

Among the phylogenetic reconstruction approaches that have attained widespread recognition, Bayesian inference has become increasingly popular, in large part due to the availability of open-source software packages such as the Bayesian evolutionary analysis by sampling trees (BEAST) software [11] and MrBayes [29]. Bayesian phylogenetic inference is based on a quantity called the posterior distribution of trees, which involves a summation over all trees and, for each tree, integration over all possible combinations of branch length and substitution model parameter values [20]. Analytical evaluation of this distribution is practically infeasible, and hence needs to be approximated using a numerical method, the most common being Markov chain Monte Carlo (MCMC). The basic idea is to construct a Markov chain that has as its state space the parameters of the statistical model and a stationary distribution that is the posterior distribution of the parameters (including the tree) [20]. While MCMC integration has revolutionized the field of phylogenetics [34], the continuously increasing size of data sets is pushing the field of statistical phylogenetics to its limits.

While promising approaches to improve MCMC efficiency have emerged recently from the field of computational statistics, such as sequential Monte Carlo (SMC; see, e.g., Doucet [10]) and Hamiltonian Monte Carlo (HMC; see, e.g., Neal [27]), these approaches do not yet find widespread use in phylogenetics. The primary difficulty in this adoption centers around the tree that encompasses both continuous and discrete random variables. Instead, considerable attention is being meted on techniques for parallelization [32] to improve phylogenetic software run-times. Obtaining sufficient samples from a Markov chain may take many iterations, due to the large number of trees that may describe the relationships of a group of species and high autocorrelation between the samples. It is therefore of critical importance to perform each iteration in a computationally efficient manner, making optimal use of the available hardware. High-performance computational libraries, such as the broad-platform evolutionary analysis general likelihood evaluator (BEAGLE) [3], can be useful tools to enable efficient use of multicore computer hardware (or even special-purpose hardware), while at the same time requiring minimal knowledge from the software user(s).

In this chapter, we first introduce the BEAGLE software library and its primary purpose, characteristics, and typical usages in Subheading 2, along with the hardware specifications of the devices

used for benchmarks in this chapter. In Subheading 3, we present computational benchmarks on the different hardware devices for a collection of data sets that are typically analyzed with models of varying complexity. Subheading 4 presents a brief overview of studies for which GPU computing capabilities were critical to analyze the data in a timely fashion. Given the increasing capabilities over hardware devices, we present an interesting avenue for further research in Subheading 5, in the form of adaptive MCMC.

2 The BEAGLE Library

BEAGLE [3] is a high-performance likelihood-calculation platform for phylogenetic applications. BEAGLE defines a uniform application programming interface (API) and includes a collection of efficient implementations for evaluating likelihoods under a wide range of evolutionary models, on graphics processing units (GPUs) as well as on multicore central processing units (CPUs). The BEAGLE library can be installed as a shared resource, to be used by any software aimed at phylogenetic reconstruction that supports the library. This approach allows developers of phylogenetic software to share any optimizations of the core calculations, and any program that uses BEAGLE will automatically benefit from the improvements to the library. For researchers, this centralization provides a single installation to take advantage of new hardware and parallelization techniques.

The BEAGLE project has been very successful in bringing hardware acceleration to phylogenetics. The library has been integrated into popular phylogenetics software including BEAST [11], MrBayes [29], PhyML [19], and GARLI [35] and has been widely used across a diverse range of evolutionary studies. The BEAGLE library is free, open-source software licensed under the Lesser GPL and available at https://beagle-dev.github.io.

2.1 Principles

2.1.1 Computing Observed Data Likelihoods

The most effective methods for phylogenetic inference involve computing the probability of observed character data for a set of taxa given an evolutionary model and phylogenetic tree, which is often referred to as the (observed data) likelihood of that tree. Felsenstein demonstrated an algorithm to calculate this probability [16], and his algorithm recursively computes partial likelihoods via simple sums and products. These partial likelihoods track the probability of the observed data descended from an internal node conditional on a particular state at that internal node.

The partial likelihood calculations apply to a subtree comprising a parent node, two child nodes, and connecting branches. It is repeated for each unique site pattern in the data (in the form of a multiple sequence alignment), for each possible character of the state space (e.g., nucleotide, amino acid, or codon), and for each

internal node in the proposed tree. The computational complexity of the likelihood calculation for a given tree is $O(p \times s^2 \times n)$, where p is the number of unique site patterns in the sequence (typically on the order of 10^2–10^6), s is the number of states each character in the sequence can assume (typically 4 for a nucleotide model, 20 for an amino-acid model, or 61 for a codon model), and n is the number of operational taxonomic units (e.g., species and alleles).

Additionally, the tree space is very large; the number of unrooted topologies possible for n operational taxonomic units is given by the double factorial function $(2n - 5)!!$ [15]. Thus, to explore even a fraction of the tree space, a very large number of topologies need to be evaluated, and hence a very great number of likelihood calculations have to be performed. This leads to analyses that can take days, weeks, or even months to run. Further compounding the issue, rapid advances in the collection of DNA sequence data have made the limitation for biological understanding of these data an increasingly computational problem. For phylogenetic inferences, the computation bottleneck is most often the calculation of the likelihoods on a tree. Hence, speeding up the calculation of the likelihood function is key to increasing the performance of these analyses.

2.1.2 Parallel Computation

Advances in computer hardware, specifically in parallel architectures, such as many-core GPUs, multicore CPUs, and CPU intrinsics (e.g., SSE and AVX), have created opportunities for new approaches to computationally intensive methods. The structure of the likelihood calculation, involving large numbers of positions and multiple states, as well as other characteristics, makes it a very appealing computational fit to these modern parallel processors, especially to GPUs.

BEAGLE exploits GPUs via fine-grained parallelization of functions necessary for computing the likelihood on a (phylogenetic) tree. Phylogenetic inference programs typically explore tree space in a sequential manner (Fig. 1, *tree space*) or with only a small number of sampling chains, offering limited opportunity for task-level parallelization. In contrast, the crucial computation of partial likelihood arrays at each node of a proposed tree presents an excellent opportunity for fine-grained data parallelism, which GPUs are especially suited for. The use of many lightweight execution threads incurs very low overhead on GPUs, enabling efficient parallelism at this level.

In order to calculate the overall likelihood of a proposed tree, phylogenetic inference programs perform a post-order traversal, evaluating a partial likelihood array at each node. When using BEAGLE, the evaluation of these multidimensional arrays is offloaded to the library. While each partial likelihood array is still

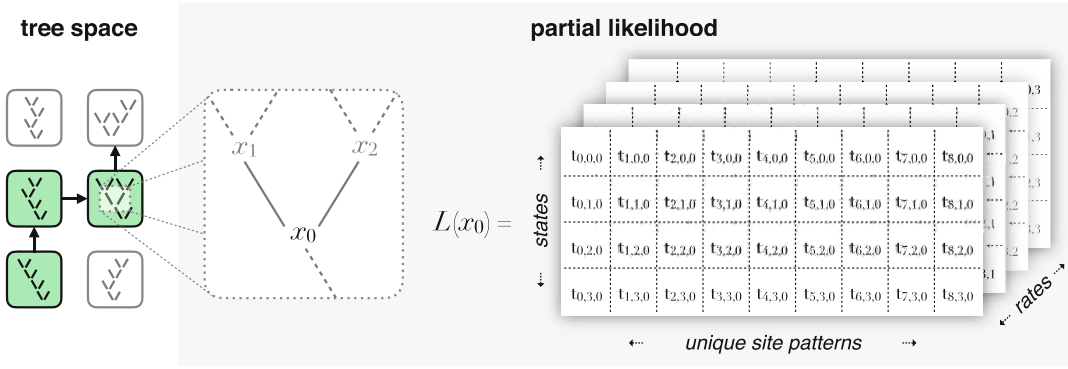


Fig. 1 Diagrammatic example of the tree sampling process and fine-grained parallel computation of phylogenetic partial likelihoods using BEAGLE for a nucleotide model problem with five taxa, nine site patterns, and four evolutionary rate categories. Each entry in a partial likelihood array *L* is assigned to a separate GPU thread *t*. In this example, 144 GPU threads are created to enable parallel evaluation of each entry of the partial likelihood array $L(x_0)$

evaluated in sequence, BEAGLE assigns the calculation of the array entries to separate GPU threads, for computation in parallel (Fig. 1, *partial likelihood*). Further, BEAGLE uses GPUs to parallelize other functions necessary for computing the overall tree likelihood, thus minimizing data transfers between the CPU and GPU. These additional functions include those necessary for computing branch transition probabilities, for integrating root and edge likelihoods, and for summing site likelihoods.

Multicore CPU parallelization through BEAGLE can only be done via multiple instances of the library, such that each instance computes a different data partition. Multiple CPU threads can be used (e.g., one for each partition) if the application program (BEAST, for the remainder of this chapter) creates the BEAGLE instances in separate computation threads, which will be the case when using BEAST. This approach suits the trend of increasingly large molecular sequence data sets, which are often heavily partitioned in order to better model the underlying evolutionary processes. BEAGLE itself does not employ any kind of load balancing nor are the site columns computed in individual threads. Each BEAGLE instance only parallelizes computation on CPUs via SSE vectorization.

BEAGLE can also use GPUs to perform partitioned analyses, however for problem sizes that are insufficiently large to saturate the capacity of one device, efficient computation requires multiple GPUs. Recent progress has been made in parallelizing the computation of multiple data subsets on one GPU [1], and future releases of BEAGLE will include this capability.

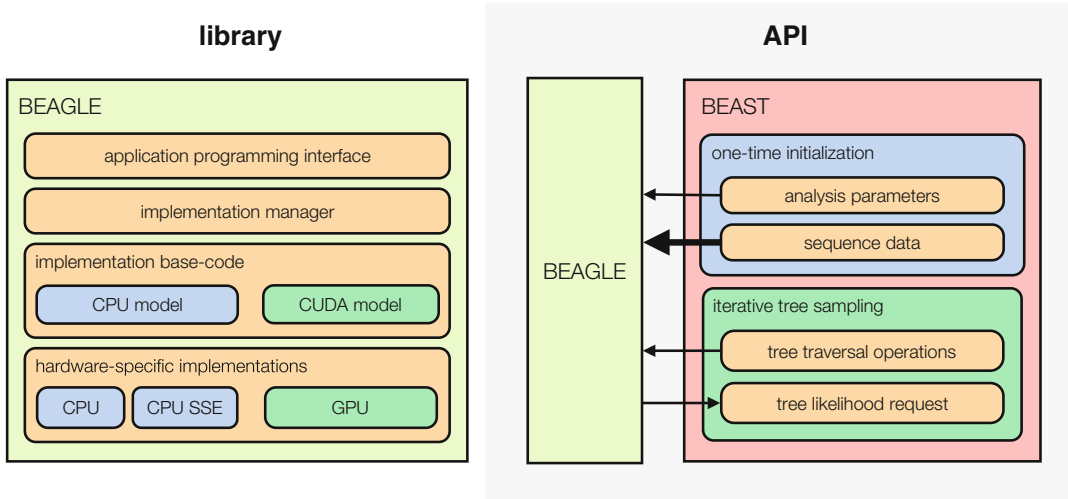


Fig. 2 Layer diagram depicting BEAGLE library organization, and illustration of API use. Arrows indicate direction and relative size of data transfers between the client program and library

2.2 Design

2.2.1 Library

The general structure of the BEAGLE library can be conceptualized as layers (Fig. 2, *library*), the upper most of which is the application programming interface. Underlying this API is an implementation management layer, which loads the available implementations, makes them available to the client program, and passes API commands to the selected implementation.

The design of BEAGLE allows for new implementations to be developed without the need to alter the core library code or how client programs interface with the library. This architecture also includes a plugin system, which allows implementation-specific code (via shared libraries) to be loaded at runtime when the required dependencies are present. Consequently, new frameworks and hardware platforms can more easily be made available to programs that use the library, and ultimately to users performing phylogenetic analyses.

Currently, the implementations in BEAGLE derive from two general models. One is a serial CPU implementation model, which does not directly use external frameworks. Under this model, there is a standard CPU implementation, and one with added SSE intrinsics, which uses vector processing extensions present in many CPUs to parallelize computation across character state values. The other implementation model involves an explicit parallel accelerator programming model, which uses the CUDA external computing framework to exploit NVIDIA GPUs. It implements fine-grained parallelism for evaluating likelihoods under arbitrary molecular evolutionary models, and thus harnessing the large number of processing cores to efficiently perform calculations [3, 32].

Recent progress has been made in developing new implementations for BEAGLE, beyond those described here, thus expanding the range of hardware that can be used. Upcoming releases of the library will include additional support for CPU parallelism via a multi-threaded implementation and will support the OpenCL standard, enabling the use of AMD GPUs [2].

2.2.2 Application Programming Interface

The BEAGLE API was designed to increase performance via fine-scale parallelization while reducing data transfer and memory copy overhead to an external hardware accelerator device (e.g., GPU). Client programs, such as BEAST [11], use the API to offload the evaluation of tree likelihoods to the BEAGLE library (Fig. 2, *API*). API functions can be subdivided into two categories: those which are only executed once per inference run and those which are repeatedly called as part of the iterative sampling process. As part of the one-time initialization process, client programs use the API to indicate analysis parameters such as tree size and sequence length, as well as specifying the type of evolutionary model and hardware resource(s) to be used. This allows BEAGLE to allocate the appropriate number and size of data buffers on device memory. Additionally at this initialization stage, the sequence data is specified and transferred to device memory. This costly memory operation is only performed once, thus minimizing its impact.

During the iterative tree sampling procedure, client programs use the API to specify changes to the evolutionary model and instruct a series of partial likelihood operations that traverse the proposed tree in order to find its overall likelihood. BEAGLE efficiently computes these operations and makes the overall tree likelihood as well as per-site likelihoods available via another API call.

2.3 Performance

Peak performance with BEAGLE is achieved when using a high-end GPU; however, the relative gain over using a CPU depends on model type and problem size as more demanding analyses allow for better utilization of GPU cores. Figure 3 shows speedups relative to serial CPU code when using BEAGLE with an NVIDIA P100 GPU for the critical partial likelihood function, with increasing unique site pattern counts and for two model types. Computing these likelihoods typically accounts for over 90% of the total execution time for phylogenetic inference programs and the relationship between speedups and problem size observed here primarily matches what would be observed for a full analysis.

Figure 3 includes performance results for computing partial likelihoods under both nucleotide and codon models. The vertical axis labels show the speedup relative to the average performance of a baseline serial, single threaded and non-vectorized, CPU implementation. This nonparallel CPU implementation provides a consistent performance level across different problem sizes and

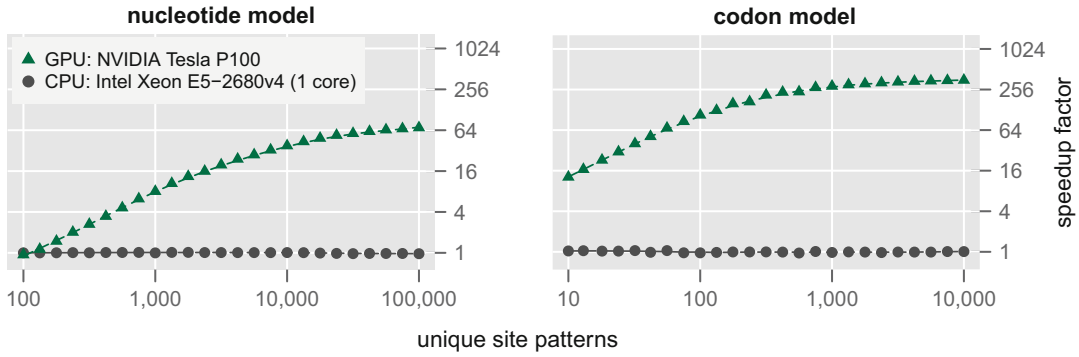


Fig. 3 Plots showing BEAGLE partial likelihood computation performance on the GPU relative to serial CPU code, under nucleotide and codon models and for an increasing number of unique site patterns. Speedup factors are on a log-scale

provides a relevant point of comparison as most phylogenetic inference software packages use serial code as their standard.

Using a nucleotide model, relative GPU performance over the CPU strongly scales with the number of site patterns. For very small numbers of patterns, the GPU exhibits poor performance due to greater execution overhead relative to overall problem size. GPU performance improves quickly as the number of unique site patterns is increased and by 10,000 patterns it is closer to a saturation point, continuing to increase but more slowly. At 100,000 nucleotide patterns, the GPU is approximately 64 times faster than the serial CPU implementation.

For codon-based models, GPU performance is less sensitive to the number of unique site patterns. This is due to the better parallelization opportunity afforded by the 61 biologically meaningful states that can be encoded by a codon. The higher state count of codon data compared to nucleotide data increases the ratio of computation to data transfer, resulting in increased GPU performance for codon-based analyses. For a problem size with 10,000 codon patterns, the GPU is over 256 times faster than the serial CPU implementation.

2.4 Memory Usage

When assessing the suitability of a phylogenetic analysis for GPU acceleration via BEAGLE, it is also important to consider if the GPU has sufficient on-board memory for the analysis to be performed. GPUs typically have less memory than what is available to CPUs, and the high transfer cost of moving data from CPU to GPU memory prevents direct use of CPU memory for GPU acceleration.

Figure 4 shows how much memory is required for problems of different sizes when running nucleotide and codon-model analyses in BEAST with BEAGLE GPU acceleration. Note that when multiple GPUs are available, BEAST can partition a data set into

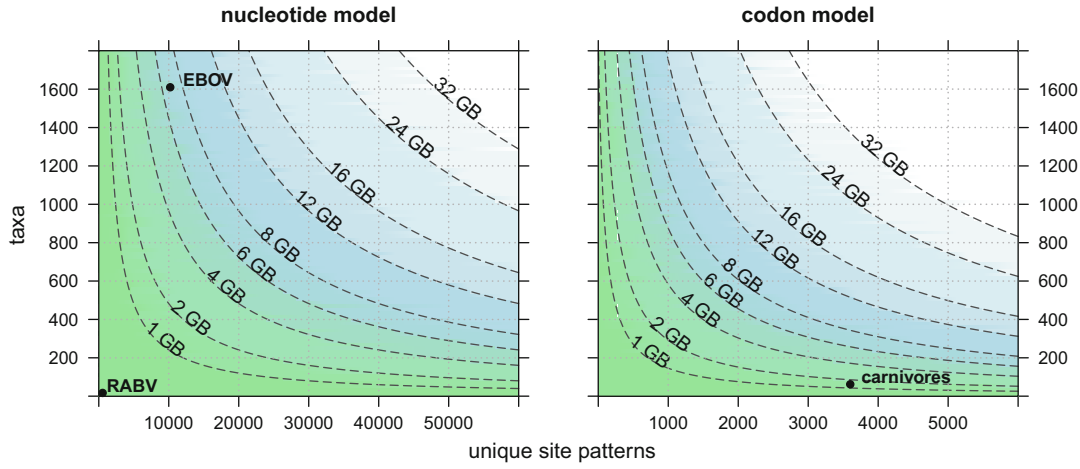


Fig. 4 Contour plots depicting BEAGLE memory usage on GPUs for BEAST nucleotide and codon-model analyses with 4 rate categories in double-precision floating-point format, for a range of problem sizes with different numbers of taxa and of unique site patterns. Memory requirements shown here assume an unpartitioned dataset. Partitioned analyses and more sophisticated models that use multiple BEAGLE instances incur memory overhead per additional library instance. Black dots indicate memory usage requirements for the unpartitioned version of three data sets subsequently described in this chapter

separate BEAGLE instances, one for each GPU. Thus, each GPU will only require as much memory as necessary for the data subset assigned to it. Typical PC-gaming GPUs have 8 GB of memory or less, while GPUs dedicated to high-performance computing, such as the NVIDIA Tesla series, may have as much as 24 GB of memory.

2.5 Hardware

Highly parallel computing technologies such as GPUs have overtaken traditional CPUs in peak performance potential and continue to advance at a faster pace. Additionally, the memory bandwidth available to the processor is especially relevant to data-intensive computations, such as the evaluation of nucleotide model likelihoods. In this measure as well, high-end GPUs significantly outperform equivalently positioned CPUs.

BEAGLE was designed to take advantage of this trend of increasingly advanced GPUs and uses runtime compilation methods to optimize code for whichever generation of hardware is being used. Table 1 lists hardware specifications for the processors used in this chapter. We note that further advancements in the GPU market for scientific computing are on its way, with NVIDIA preparing the launch (at the time of writing) of the Tesla V100 in Q3 of 2017. The new NVIDIA Tesla V100 features a total of 5120 CUDA cores and comes equipped with 32 GB of on-board memory with 900 GB/s of bandwidth. As such, it seems to have the potential to reach 7.5 TFLOPs in double-precision peak performance (DP PP), a roughly 50% increase over their current flagship, the Tesla P100.

Table 1
Hardware specifications for the Intel CPUs and NVIDIA GPUs used in this chapter

Hardware	Year	Cores	Memory	Bandwidth	DP PP
Xeon E5-2680v2	2013	2 × 10	64 GB	60 GB/s	0.45 TFLOPS
Xeon E5-2680v3	2014	2 × 12	64 GB	68 GB/s	0.96 TFLOPS
GTX 590	2011	2 × 512	2 × 1.5 GB	164 GB/s	0.31 TFLOPS
Tesla K20X	2012	2688	6 GB	250 GB/s	1.31 TFLOPS
Tesla K40	2013	2880	12 GB	288 GB/s	1.43 TFLOPS
Quadro P5000	2016	2560	16 GB	288 GB/s	NA
Tesla P100	2016	3584	16 GB	720 GB/s	4.70 TFLOPS

Estimated performance in double-precision peak performance (DP PP) taken from the manufacturer's website. Note that the Quadro P5000 GPU only lists performance in single precision and we hence list its double-precision performance as not available (NA)

3 Results

In this section, we compare the performance of various typical Bayesian phylogenetic, phylogenomic, and phylodynamic analyses on different multicore architectures. In Subheading 3.1, we analyze a data set of mitochondrial genomes [32] using a high-dimensional model of codon substitution which, albeit low in number of parameters, is particularly challenging in phylogenetic analyses specifically because of the high-dimensional state space. In Subheading 3.2, we analyze the largest Ebola virus data set at the time of publication [12] using a collection of nucleotide substitution models, i.e., one per codon position and an extra one for analyzing the intergenic regions, where the large number of taxa and unique site patterns offer an interesting test case for the comparison between CPU and GPU performance. Finally, Subheading 3.3 reports on the performance of analyzing data sets that complement sequence data with discrete trait data (typically host data or geographic data), for which transition rates between (a potential large number of) discrete trait states are parameterized as a generalized linear model (GLM). All performance evaluations in this results section were run for 100,000 iterations (which is usually insufficient to achieve convergence) in BEAST v1.8.4 [11], using double precision (both on CPU and GPU) and in conjunction with BEAGLE v2.1.2 [3]. By default, BEAST—through BEAGLE—uses SSE2 (Streaming SIMD Extensions 2), an SIMD instruction set extension to the x86 architecture, when performing calculations on CPU.

3.1 Carnivores

Selection is a key evolutionary process in shaping genetic diversity and a major focus of phylogenomics investigations [23]. Researchers frequently evaluate the strength of selection operating on genes or even individual codons in the entire phylogeny or in a subset of branches using statistical methods. Codon substitution models have been particularly useful for this purpose because they allow estimating the ratio of non-synonymous and synonymous substitution rates (dN/dS) in a phylogenetic framework. Goldman and Yang [18] and Muse and Gaut [26] developed the first codon-based evolutionary models (GY and MG, respectively), i.e., models that have codons as their states, incorporating biologically meaningful parameters such as transition/transversion bias, variability of a gene, and amino acid differences.

Full codon substitution models are computationally expensive compared to standard nucleotide substitution models due to their large state space. Compared to nucleotide models (4 states) and amino acid models (20 states), a full vertebrate mitochondrial codon model has 60 states (ignoring the four nonsense or stop codons). We restrict ourselves to the standard GY codon substitution model implementation in BEAST [11], employ the standard assumption that mutations occur independently at the three codon positions and therefore only consider substitutions that involve a single-nucleotide substitution, and assume that codons evolve independently from one another. Additionally, we allow for substitution rate heterogeneity among codons using a discrete gamma distribution (i.e., each codon is allowed to evolve at a different substitution rate) [33], which increases the computational demands of such an analysis fourfold (given that we allow for the standard assumption of four discrete rate categories).

As a first application of using state-of-the-art hardware in statistical phylogenetics, we reevaluate the performance of a full codon model on a set of mitochondrial genomes from extant carnivores and a pangolin outgroup [4, 32]. This genomic sequence alignment contains 10,869 nt columns that code for 12 mitochondrial proteins and when translated into a single 60-state vertebrate mitochondrial codon model, yields a total of 3623 alignment columns, of which 3601 site patterns are unique [32]. Figure 5 shows a comparison of the computational throughput between various CPU and GPU computing platforms. To this end, we make use of an option in BEAST [11] to split an alignment into two or more pieces of equal length, with each resulting alignment being evaluated on a separate processor core or computing device for optimal performance. Figure 5 shows that the analysis scales remarkably well on CPU, where the use of each additional processor core results in a performance increase. This can be attributed to the use of full codon models, which invokes a higher workload when evaluating each likelihood and hence more concurrent evaluation compared to thread communication. As the evaluation of the total

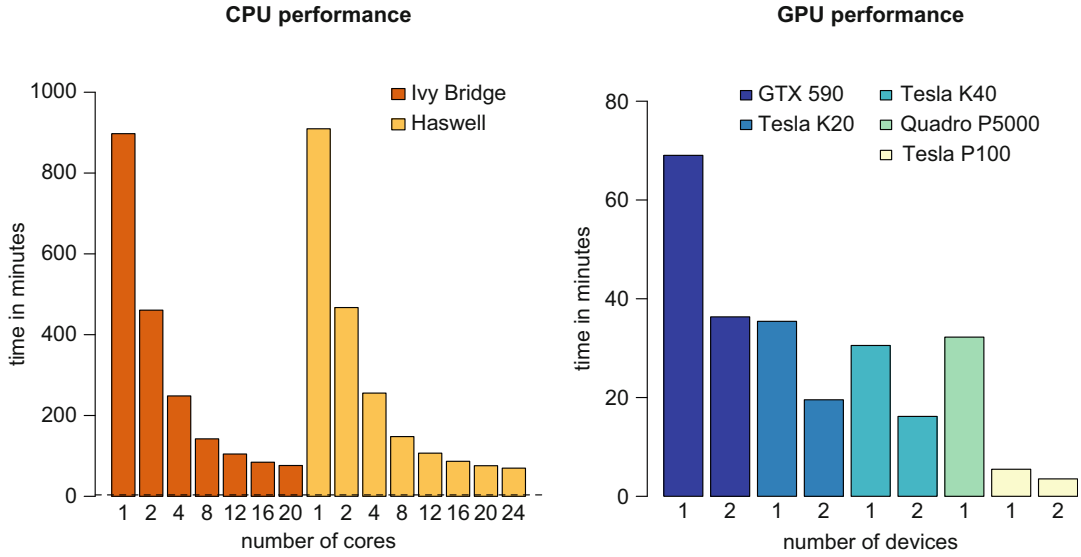


Fig. 5 Performance evaluation of different CPU and GPU configurations when estimating a single full codon substitution model on a full genome mitochondrial data set. Left: performance comparison on multicore CPUs (with the dashed line indicating optimal GPU performance). Right: performance comparison on GPUs. The numbers below the bars indicate the number of partitions/threads used in each analysis. The GTX 590, a GPU released in 2011 for PC gaming, performs equally well as a 24-core CPU configuration. However, GPUs from the Tesla generation, aimed at scientific computing, drastically outperform the CPU system, with the recently released P100 showcasing the impressive improvements in the GPU market, running 166 times faster than a single-core CPU setup

amount of unique site patterns is split over more processor cores, the workload per core decreases and the communication overhead increases, resulting in smaller relative performance increases.

The performance of a 24-core CPU setup is easily matched by a single GPU (the GTX 590) that was originally aimed at the market of PC gaming. However, subsequent improvements in GPU cards for scientific computing have yielded impressive performance gains, with a single Tesla K20 GPU outperforming 2 GTX 590 GPUs. Whereas the advent of the Tesla K40 offered further performance increases, it was mainly welcomed for having twice the amount of on-board memory, allowing for much larger data sets to be analyzed on GPU. The recent introduction of the Tesla P100 GPU promised and delivered astonishing results, as shown in Fig. 5, with a single Tesla P100 GPU delivering six times the performance of a Tesla K40 GPU on these high-dimensional full codon models. We conclude that the use of a high-performance computational library such as BEAGLE, in combination with a powerful GPU, has significantly facilitated the evaluation of and phylogenetic inference with full codon models.

3.2 Ebola Virus

The original developments within the BEAGLE library offered considerable computational speedup when evaluating codon models—up to a 52-fold increase when employing three GPU cards—and nucleotide models—up to a 15-fold increase when using three GPU cards—in double precision [32]. This may have resulted in the perception that GPU cards are mainly useful when evaluating codon models, but that the benefit for fitting models was not sufficiently substantial to warrant GPUs. To offer an objective assessment of the usefulness of GPUs in such cases, we analyze the use of various CPU and GPU configurations on a full genome Ebola virus data set, consisting of 1610 publicly available genomes sampled over the course of the 2013–2016 Ebola virus disease epidemic in West Africa [12] (we discuss this study in more detail in Subheading 4). This data set encompassing 18,992 nt columns is modelled with four partitions: one for each codon position and one additional partition for the intergenic region (which consists of several noncoding regions interspersed in the genome). The three codon partitions contain, respectively, 2366, 2328, and 2731 unique site patterns, while the intergenic partition contains 2785 unique site patterns. We model among-site rate variation [33] in each partition independently, which confronts us with a computationally demanding analysis for this large number of taxa and unique site patterns.

Figure 6 shows how the performance of such a large nucleotide data set scales with the available CPU and GPU resources. Contrary to the carnivores data set analysis in Subheading 3.1, this analysis does not scale particularly well with the number of CPU cores available, as the main benefit lies with splitting each partition into two subpartitions and only limited performance gains can be observed when using additional partitions or threads. Popular single GPU cards for scientific computing—such as the Tesla K40—match the optimal performance brought about by using 16 CPU cores, and may provide a useful alternative to multicore CPU systems. However, the decreasing cost for increasingly parallel multicore CPU systems makes this a difficult matchup for slightly older GPUs. More recently introduced GPU cards, such as the Tesla P100, are able to deliver a substantial performance improvement over a multicore CPU setup, with two Tesla P100 GPUs running in parallel offering over twice the performance of a 16-core CPU setup. We note that the GTX 590 cards, as well as a single Tesla K20 card, do not contain sufficient on-board memory to hold the full data set and as such, these benchmarks could not be run on those resources.

3.3 Phylogeography

As shown in the results in Subheadings 3.1 and 3.2, different partitions of the aligned sequence data can contain a large number of unique site patterns, rendering phylogenomic inference challenging. However, other data types are also included more

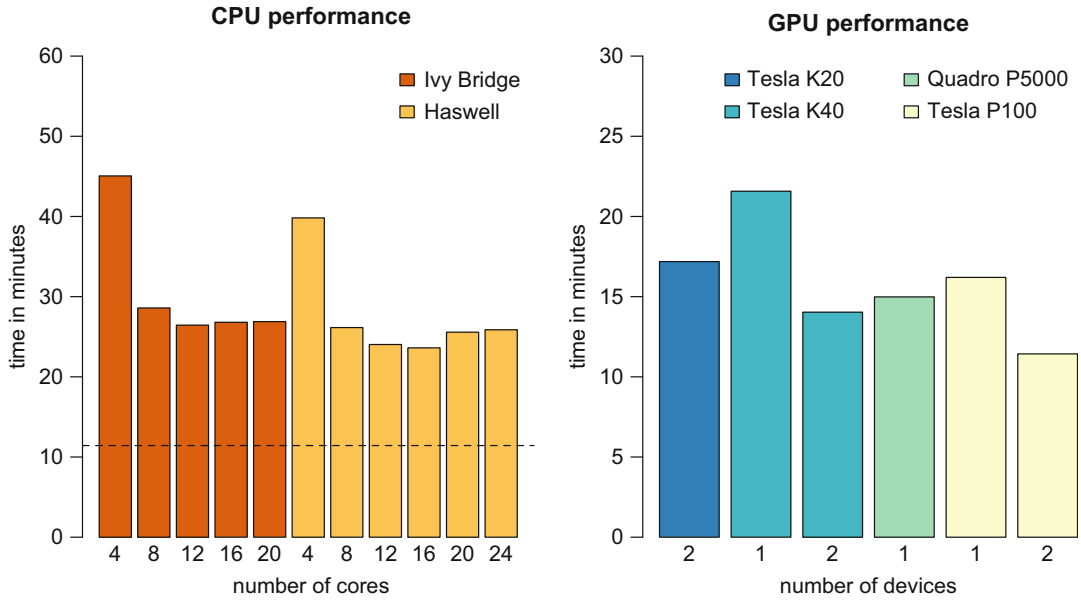
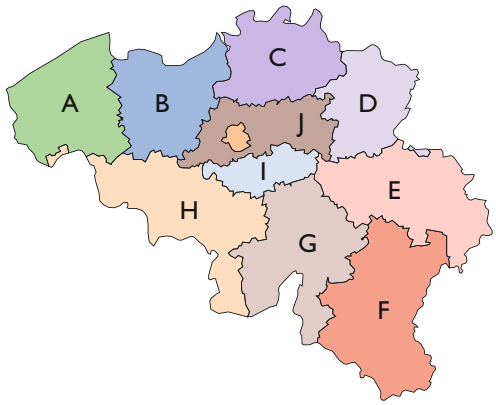


Fig. 6 Performance evaluation of different CPU and GPU configurations when estimating a four-partition nucleotide substitution model on a full genome Ebola virus data set. Left: performance comparison on multicore CPUs (with the dashed line indicating optimal GPU performance). Right: performance comparison on GPUs. The numbers below the bars indicate the number of partitions/threads used in each analysis. Because of the lower dimensionality compared to full codon models, nucleotide substitution models currently take less advantage of the large amounts of cores present on GPUs. The fastest GPU configuration—consisting of two Tesla P100 GPUs—outperforms by 107% the fastest CPU configuration that employs 16 threads on a 24-core Haswell CPU

frequently, such as trait data to be analyzed alongside the sequence data and hence potentially influencing the outcome of such an analysis (for an overview, see, e.g., Baele et al. [6]).

Arguably, the most frequently considered traits in phylodynamics, and molecular evolution in general, are spatial locations. The interest in spatial dispersal has developed into its own research field referred to as phylogeography, with Bayesian inference of discrete phylogenetic diffusion processes being adopted in the field of biogeography [30]. Jointly estimating the phylogeny and the trait evolutionary process, Lemey et al. [23] implemented a similar Bayesian full probabilistic connection between sequences and traits in BEAST [11], with applications focusing on spatiotemporal reconstructions of viral spread. These approaches offer extensive modelling flexibility at the expense of a quadratic growth in number of instantaneous rate parameters in the continuous-time Markov chain (CTMC) model as a function of the state dimensionality of the trait. This can be seen in Fig. 7, which shows two maps with different numbers of (discrete) locations and the corresponding CTMC models that describe the instantaneous rates of transition between these locations.

	A	B	C
A	.	λ_{AB}	λ_{AC}
B	λ_{BA}	.	λ_{BC}
C	λ_{CA}	λ_{CB}	.



	A	B	C	D	E	F	G	H	I	J
A	.	λ_{AB}	λ_{AC}	λ_{AD}	λ_{AE}	λ_{AF}	λ_{AG}	λ_{AH}	λ_{AI}	λ_{AJ}
B	λ_{BA}	.	λ_{BC}	λ_{BD}	λ_{BE}	λ_{BF}	λ_{BG}	λ_{BH}	λ_{BI}	λ_{BJ}
C	λ_{CA}	λ_{CB}	.	λ_{CD}	λ_{CE}	λ_{CF}	λ_{CG}	λ_{CH}	λ_{CI}	λ_{CJ}
D	λ_{DA}	λ_{DB}	λ_{DC}	.	λ_{DE}	λ_{DF}	λ_{DG}	λ_{DH}	λ_{DI}	λ_{DJ}
E	λ_{EA}	λ_{EB}	λ_{EC}	λ_{ED}	.	λ_{EF}	λ_{EG}	λ_{EH}	λ_{EI}	λ_{EJ}
F	λ_{FA}	λ_{FB}	λ_{FC}	λ_{FD}	λ_{FE}	.	λ_{FG}	λ_{FH}	λ_{FI}	λ_{FJ}
G	λ_{GA}	λ_{GB}	λ_{GC}	λ_{GD}	λ_{GE}	λ_{GF}	.	λ_{GH}	λ_{GI}	λ_{GJ}
H	λ_{HA}	λ_{HB}	λ_{HC}	λ_{HD}	λ_{HE}	λ_{HF}	λ_{HG}	.	λ_{HI}	λ_{HJ}
I	λ_{IA}	λ_{IB}	λ_{IC}	λ_{ID}	λ_{IE}	λ_{IF}	λ_{IG}	λ_{IH}	.	λ_{IJ}
J	λ_{JA}	λ_{JB}	λ_{JC}	λ_{JD}	λ_{JE}	λ_{JF}	λ_{JG}	λ_{JH}	λ_{JI}	.

Fig. 7 Graphical depiction of the Benelux countries (top) and the provinces of Belgium (bottom). When these countries and provinces are used as discrete location states in a discrete trait model, this yields, respectively, a 3 $\times$ 3 and a 10 $\times$ 10 CTMC model with instantaneous rates of transition between each pair of locations. Such models are subject to the same restrictions as those used in popular substitution models, i.e., the rows sum to 0. As such, these CTMC models consist of, respectively, 6 and 90 free parameters to be estimated

Many phylodynamic hypotheses can be addressed through the combination of genetic and trait data, but additional data in the form of covariates can help explain the evolutionary or epidemiological process. Such covariates can be used in a GLM formulation on a matrix of transition rate parameters between locations defining a CTMC process. Lemey et al. [25] developed an approach to simultaneously reconstruct spatiotemporal history and identify which combination of covariates associates with the pattern of spatial spread. This approach involves parameterizing each rate of among-location movement, typically denoted as the ijth elements (λ_{ij}) of the CTMC transition rate matrix, in the phylogeographic model as a log linear function of various potential covariates:

$$\log\lambda_{ij} = \beta_1\delta_1 x_{i,j,1} + \beta_2\delta_2 x_{i,j,2} + \cdots + \beta_N\delta_N x_{i,j,N}, \quad (1)$$

where β_i is the estimated effect size of covariate x_i, δ_i is a binary indicator that tracks the posterior probability of the inclusion of

covariate x_i in the model, and N equals the number of covariates; further, in the case of Fig. 7: $i, j \in \{A, B, C\}$ with $N = 3$ (top), and $i, j \in \{A, B, C, D, E, F, G, H, I, J\}$ with $N = 10$ (bottom). Priors and posteriors for the inclusion probabilities (δ) can be used to express the support for each predictor in terms of Bayes factors (for more information, *see* Baele et al. [6], Lemey et al. [25]). We discuss examples of such possible predictors in a phylogeographic setting in Subheading 4 but focus here on performance benchmarks for such generalized linear models.

3.3.1 Bat Rabies

We here assess the performance of a phylodynamic setup aimed at reconstructing the spatial dispersal and cross-species dynamics of rabies virus (RABV) in North American bat populations based on a set of 372 nucleoprotein gene sequences (nucleotide positions: 594–1353). The data set comprises a total of 17 bat species sampled between 1997 and 2006 across 14 states in the USA [31]. Two additional species that had been excluded from the original analysis owing to a limited amount of available sequences, *Myotis austroriparius* (Ma) and *Parastrellus hesperus* (Ph), are also included here [14]. We also include a viral sequence with an unknown sampling date (accession no. TX5275, sampled in Texas from *Lasiurus borealis*), which will be adequately accommodated in our inference. This leads to a total of 548 unique site patterns. Following Faria et al. [14], we employ two GLM-diffusion models for this analysis, one on the discrete set of 17 bat species and another on the discrete set of 14 location states.

Figure 8 shows the performance of various multicore platforms on the bat rabies Bayesian phylodynamic analysis. In contrast to previous examples, the low number of sites (and hence unique site patterns) in the alignment does not offer many options for splitting the observed data likelihood over additional threads. While four CPU cores offer the optimal performance across our CPU platforms, using more threads for the analysis causes serious communication overhead, slowing down the analysis. Comparing the CPU results with the GPU results shows that, across all multicore platforms tested, a 4-core CPU offers the best performance.

Nonetheless, this scenario provides a very interesting use case for employing multiple graphics cards for scientific computing. Even though the (relatively small) dimensions of this particular example do not allow for a performance increase, it will be beneficial for higher-dimensional cases to compute each diffusion model on a separate GPU. When assuming independent diffusion processes that only depend on the underlying phylogeny, each of the trait diffusion models can be computed on a different GPU, whereas the data alignment can be split into two subpartitions of equal complexity (i.e., with an equal number of unique site patterns) and hence also be computed in parallel over the two GPUs. However, the limited sequence data size and the relatively restricted

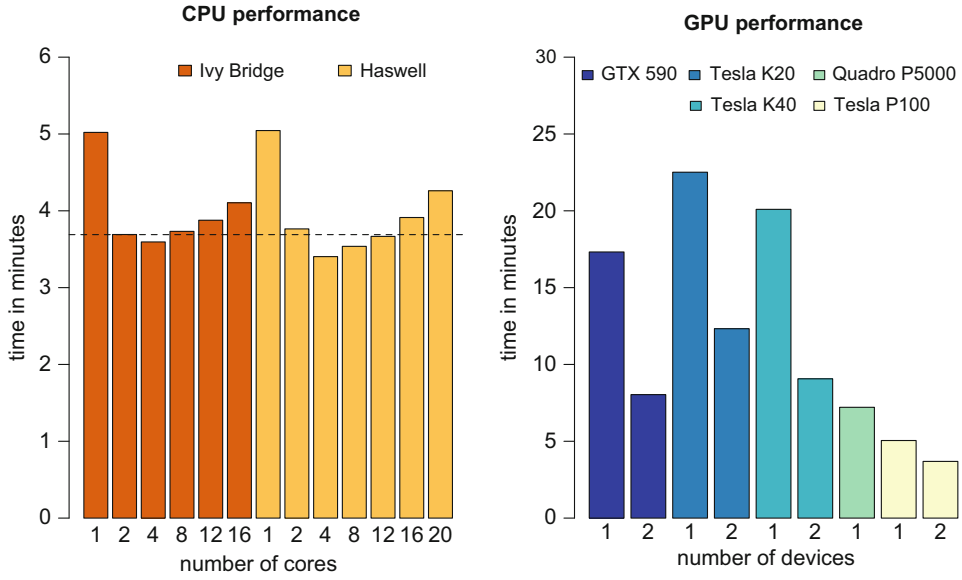


Fig. 8 Performance evaluation of different CPU and GPU configurations when estimating a single-partition nucleotide substitution model on a rabies virus data set, along with two GLMs. Left: performance comparison on multicore CPUs (with the dashed line indicating optimal GPU performance). Right: performance comparison on GPUs. Given the short length of the alignment, communication overhead becomes quite severe when adding a large number of CPU processor cores. Even two state-of-the-art Tesla P100 GPUs fail to outperform four CPU cores, given the low number of sequences and the low number of states for the GLMs

number of discrete locations make this data set less suited for illustrating performance increases using GPUs.

3.3.2 *Ebola Virus*

We here assess the performance of a similar setup as in the previous section, but using the data from the 2013–2016 West African epidemic caused by the Ebola virus [12]. We hence use the nucleotide data from the previous Ebola example (*see* Subheading 3.2) and augment it with location states. Using a phylogeographic GLM that integrates covariates of spatial spread, we have examined which features influenced the spread of EBOV among administrative regions at the district (Sierra Leone), prefecture (Guinea), and country (Liberia) levels. This resulted in a GLM parameterizing transition rates among 56 discrete location states according to 25 potential covariates (*see* Dudas et al. [12] for more information), resulting in a computationally challenging analysis.

As shown in Fig. 9, we have evaluated the performance of this challenging data set on our different multicore platforms. By comparing these benchmarks with those in Fig. 6, it's clear that the addition of a high-dimensional discrete trait model is much harder to process for any multicore CPU configuration. Adding more CPU cores to the analysis does not improve performance by much, indicative of the discrete trait model being the main

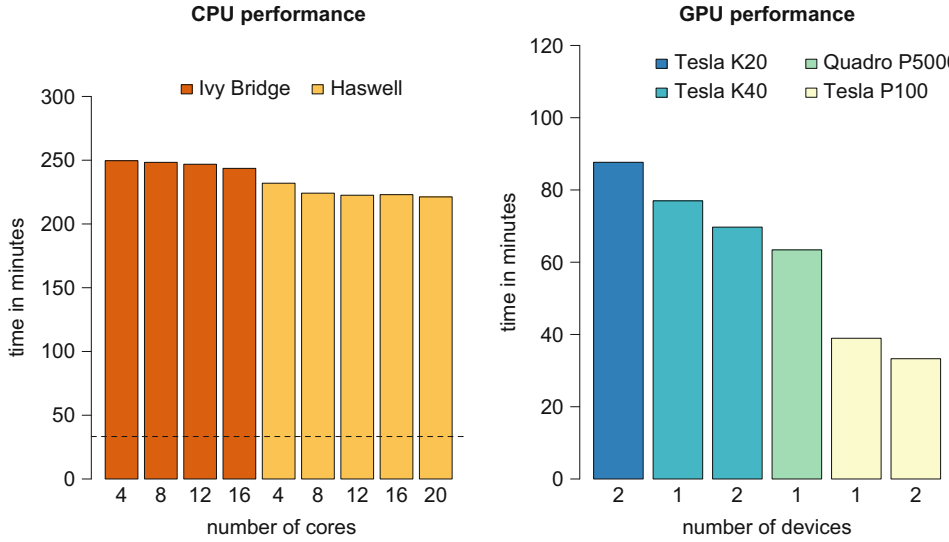


Fig. 9 Performance evaluation of different CPU and GPU configurations when estimating a four-partition nucleotide substitution model on a full genome Ebola virus data set along with a GLM-diffusion model for the location traits. Left: performance comparison on multicore CPUs (with the dashed line indicating optimal GPU performance). Right: performance comparison on GPUs. While a previous comparison of solely the nucleotide data resulted in a performance increase of over 100% for GPUs over CPUs, we observe a far more pronounced benefit for GPU in this case with two Tesla P100 GPUs outperforming a 12-core CPU setup by 568%

bottleneck in this analysis. This can be attributed to the high dimension of the discrete phylogeographic model [23] and the fact that this model describes a single column of characters, which cannot be split into multiple partitions. Relative to the computational complexity of modelling the location states, splitting the observed sequence data over multiple partitions/threads yields relatively small performance improvements.

Some of the (older) GPUs cannot fit the full data set in memory (such as the GTX 590 and a single Tesla K20), but those that are able to vastly outperform any CPU setup. Further, as these GPUs are better equipped to handle high-dimensional models, splitting the observed sequence data over multiple physical cards still yields noticeable performance gains. In contrast to Fig. 8, where two discrete phylogeographic models were used that could each be computed on different GPUs, the fact that this example only considers a single trait observation explains why less performance gains can be obtained by adding an additional GPU to the analysis.

4 Examples

In this section, we highlight examples of large sequence data sets that are augmented with trait data in the form of discrete geographic locations, for which BEAGLE [3] offers impressive

computational benefits, specifically when running these analyses on powerful graphics cards for scientific computing. Further, discrete phylogeographic models can be equipped with generalized linear models to identify predictors of pathogen spread. Both inclusion probabilities and conditional effect sizes for these predictors are estimated in order to determine support for such explanatory variables of (pathogen) spread.

4.1 Human Influenza H3N2

A potentially powerful predictor for the behavior of influenza and other infectious diseases comes in the form of information on global human movement patterns, of which the worldwide air transportation network is by far the best studied system of global mobility in the context of human infectious diseases [8].

Lemey et al. [25] use a discrete phylogeographic model equipped with a GLM to show that the global dynamics of influenza H3N2 are driven by air passenger flows, whereas at more local scales spread is also determined by processes that correlate with geographic distance. For a data set that encompasses 1441 time-stamped hemagglutinin sequences (sampled between 2002 and 2007) and up to 26 locations to be used in a discrete phylogeographic model equipped with a GLM, BEAGLE can offer substantial performance gains. A snapshot of a visual reconstruction through geographic space is presented in Fig. 10, which includes a summary of the support for the collection of covariates in the

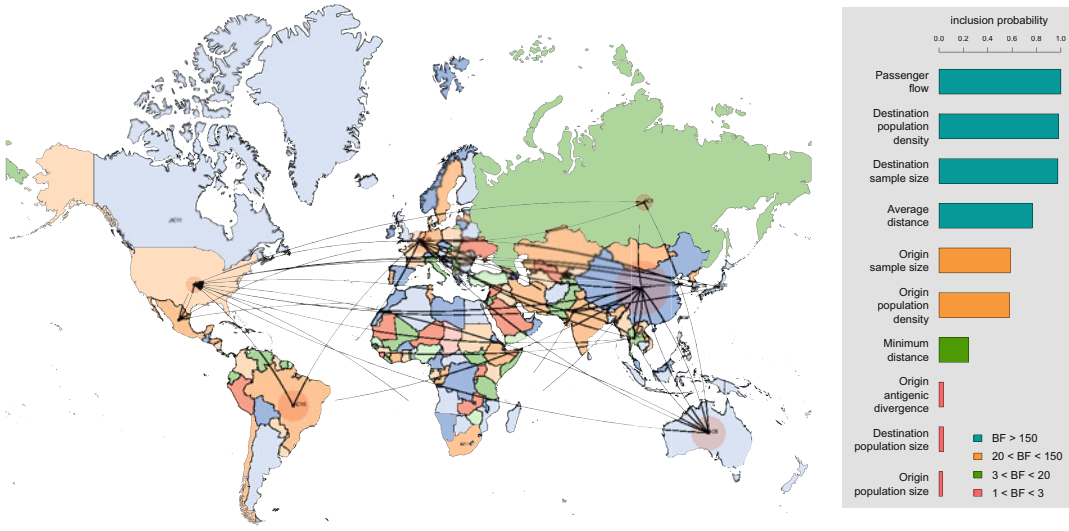


Fig. 10 Snapshot of the geographic spread of human influenza subtype H3N2, based on 1441 hemagglutinin sequences sampled between 2002 and 2007 [25]. A discrete phylogeographic approach was used, allocating the sequence data into a discrete number of locations and employing a generalized linear model on the parameters that model geographic spread. Inclusion probabilities and Bayes factor support are shown for the most prominent predictors of H3N2 geographic spread. D3 visualization is made using SpreaD3 [7], with circular polygon areas proportional to the number of tree lineages maintaining that location at that time

GLM that offer the strongest contribution to spatial spread among those tested. As illustrated in Fig. 10 (but *see* Lemey et al. [25] for additional data), there is strong evidence that air passenger flow is among the most dominant drivers of the global dissemination of H3N2 influenza viruses. Further, geographic spread is found to be inversely associated (data not shown; but *see* Lemey et al. [25]) with geographical distance between locations and with origin and destination population densities, which may seem counterintuitive. As the authors state, this negative association of population density with viral movement may suggest that commuting is less likely, per capita, to occur out of, or into, dense subpopulations.

4.2 Ebola Virus

During the two and a half years Ebola virus (EBOV) circulated in West Africa, it caused at least 28,646 cases and 11,323 deaths. As mentioned in Subheading 3.3.2, Dudas et al. [12] used 1610 genome sequences collected throughout the epidemic, representing over 5% of recorded Ebola virus disease (EVD) cases to reconstruct a detailed phylogenetic history of the movement of EBOV within and between the three most affected countries. This study considers a massive time-stamped data set that allows to uncover regional patterns and drivers of the epidemic across its entire duration, whereas individual studies had previously focused on either limited geographical areas or time periods. The authors use the phylogeographic GLM to test which features were important in shaping the spatial dynamics of EVD during the West African epidemic (*see* Fig. 11).

The phylogeographic GLM allowed Dudas et al. [12] to determine the factors that influenced the spread of EBOV among administrative regions at the district (Sierra Leone), prefecture (Guinea), and country (Liberia) levels. The authors find that EBOV tends to disperse between geographically close regions, with great circle distances having among the strongest Bayes factor support for inclusion in the GLM among all covariates tested (along with four other predictors). Additionally, both origin and destination population sizes are equally strongly and positively correlated with viral dissemination (*see* Fig. 11). Dudas et al. [12] conclude that the combination of the positive effect of population sizes with the inverse effect of geographic distance implies that the epidemic's spread followed a classic gravity-model dynamic, with intense dispersal between larger and closer populations. Finally, the authors found a significant propensity for virus dispersal to occur within each country, relative to internationally, suggesting that country borders may have provided a barrier for the geographic spread of EBOV.

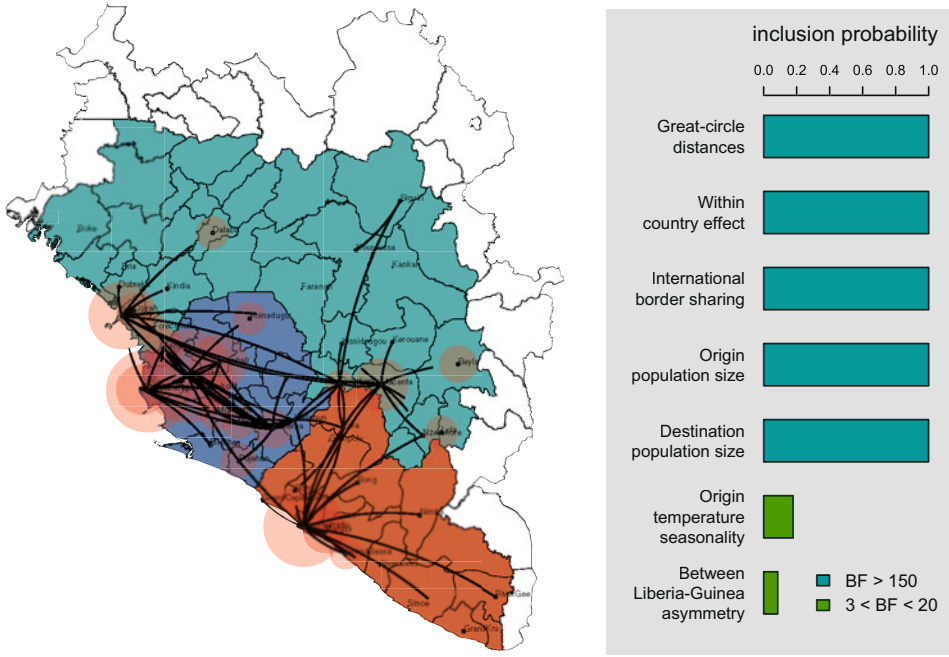


Fig. 11 Snapshot of the geographic spread of the Ebola virus during the 2013–2016 West African epidemic, based on 1610 whole genome sequences [12]. A discrete phylogeographic approach was used, allocating the sequence data into a discrete number of locations and employing a generalized linear model on the parameters that model geographic spread. Inclusion probabilities and Bayes factor support are shown for the most prominent predictors of Ebola virus geographic spread. D3 visualization is made using SpreaD3 [7], with circular polygon areas proportional to the number of tree lineages maintaining that location at that time

5 Adaptive MCMC

The various data sets described in this chapter so far have shown the use and computational performance of a wide range of models in phylogenetic, phylogenomic, and phylodynamic research. Whether employing full codon models (e.g., the carnivores data set), codon partition models (e.g., the Ebola data set), or discrete phylogeographic models, the number of parameters of a typical Bayesian phylogenetic analysis has increased drastically over the years. This is exacerbated by the use of partitioning strategies, resulting also in a potentially large array of likelihoods that need to be evaluated simultaneously, increasing run times for most phylogenetic analyses. In a similar fashion, computational resources available to researchers have also markedly increased, both in the form of multicore CPU technology and increasingly powerful graphics cards targeted towards scientific computing. The ubiquitous availability of multiprocessor and multicore computers practically has motivated the design of novel parallel algorithms to make efficient use of these machines [22, 32].

Many Bayesian phylogenetics software packages, such as BEAST [11] and MrBayes [29], do not fully exploit the inherent parallelism of such multicore systems when analyzing partitioned data because they typically update one single parameter at a time (a practice called single-component Metropolis–Hastings; Gilks et al. [17]). Such a single parameter often belongs to an evolutionary model for a single data partition, leading to only one of the potentially large collection of (observed) data likelihoods to be modified at any one time. Such a strategy does not use the computational power of modern-day multiprocessor and multicore systems to its full advantage. Updating all the models' parameters at once however leads to multiple data likelihoods being modified simultaneously, thereby making better use of the resources offered by these multicore systems (*see* Fig. 12).

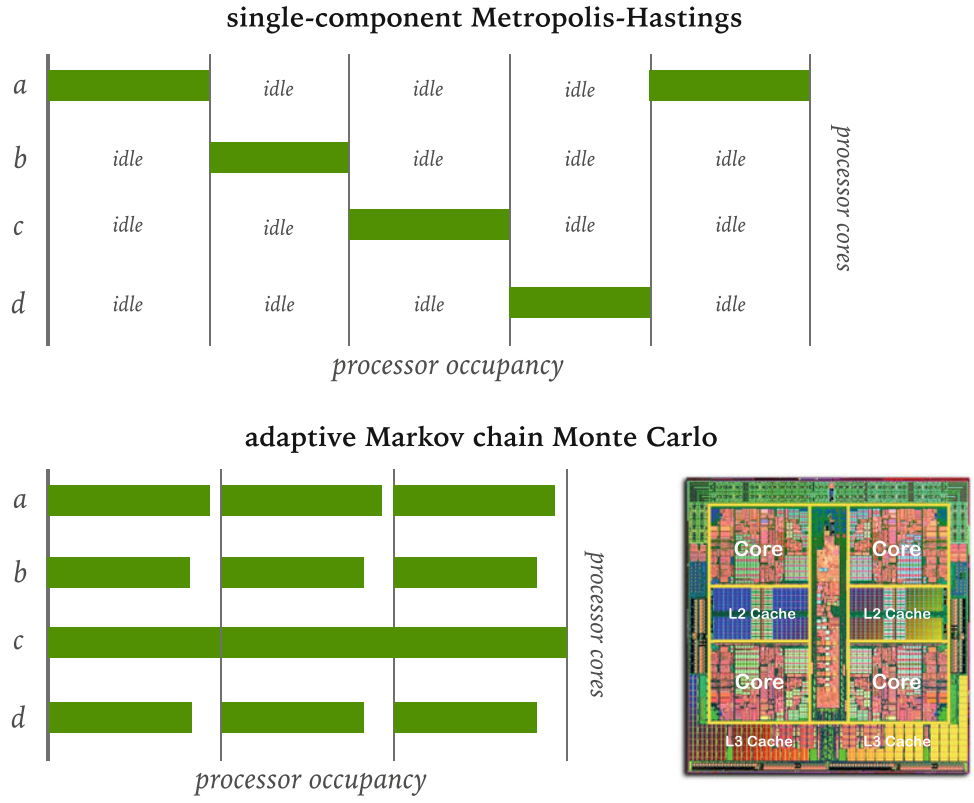


Fig. 12 Conceptual visualization on the potential benefits of an adaptive MCMC algorithm over single-component Metropolis–Hastings (green bars indicate that a processor is computing a specific likelihood). In Bayesian phylogenetics, the common practice of updating a single parameter (i.e., either *a*, *b*, *c*, or *d*) at a time leaves many CPU cores idle, underusing the computational performance of such architecture. Adaptive MCMC allows to update a collection of continuous parameters simultaneously (i.e., *a*, *b*, *c*, and *d*), putting many cores (in this case: 4) to work in a parallel fashion. Quad-Core AMD Opteron processor silicon die is shown, courtesy of Advanced Micro Devices, Inc. (AMD), obtained from Wikimedia Commons

In recent work, Baele et al. [5] propose to use multivariate components to update blocks of parameters, leading to acceptance or rejection for all of those parameters simultaneously, rather than updating all the parameters one by one in a sequential fashion using low-dimensional or scalar components [17]. To this end, the authors developed an adaptable variance multivariate normal (AVMVN) transition kernel for use in Bayesian phylogenetics, based on the work of Roberts and Rosenthal [28], to simultaneously estimate a large number of partition-specific parameters. Baele et al. [5] implemented this adaptive MCMC approach in the popular open-source BEAST software package [11], which enables this transition kernel to exploit the computational routines within the BEAGLE library [3]. The authors applied this transition kernel to a collection of clock model parameters, speciation model parameters, coalescent model parameters, and partition-specific evolutionary model parameters (which include substitution model parameters, varying rates across sites parameters, and relative rate parameters), although this kernel may find its use on parameters in many additional models.

Baele et al. [5] show that such an AVMVN transition kernel tremendously increases estimation performance over a standard set of single-parameter transition kernels. Importantly, the use of an AVMVN transition kernel requires a paradigm shift in assessing performance of transition kernels in MCMC. It is common to judge the performance of Bayesian phylogenetic software packages strictly by the time they take to evaluate proposed parameter values, often expressed in time per number of states or iterations. However, comparing transition kernels that only require a single processor core to evaluate a proposed value against transition kernels that require a collection of processor cores to evaluate a collection of proposed values simultaneously is unfair, as the latter will logically take up more time as this involves more (computational) work (on multiple processor cores). Hence, a fair comparison involves calculating the effective sample size (ESS) per time unit, as this takes into account differences in execution speed while still reporting a main statistic of interest.

We note that the approach of Baele et al. [5] has been shown to yield performance increases on CPU, but that it still needs to be tested on GPU. This is due to the specific design of the BEAGLE library [3], which evaluates a collection of BEAGLE likelihoods/instances sequentially on GPU. Current work is underway to simplify the run time process of the BEAGLE library on GPU, allowing for simultaneous evaluation of such a collection of instances.

6 Conclusions

In this chapter, we have focused on the computational challenges associated with typical analyses in the fields of phylogenetics, phylogenomics, and phylodynamics. We have provided a detailed description of how the BEAGLE library can employ multicore hardware to perform efficient likelihood evaluations and have focused on its interaction with the BEAST software package. Using benchmarks collected on a range of multicore hardware, both from the CPU and GPU market, we have shown that employing the BEAGLE high-performance computational library can considerably decrease computation time on these different systems and this for data sets with different characteristics in terms of size and complexity. The BEAGLE library allows to simultaneously compute the likelihoods for different data partitions on different CPU cores or even on different hardware devices, such as multiple GPU cards. In addition, existing data partitions can be split into multiple subpartitions to be computed in parallel across multicore hardware, yielding potentially drastic performance increases as shown in the benchmarks discussed in this chapter.

Having employed the BEAGLE library on state-of-the-art multicore hardware for a range of commonly used evolutionary models, we conclude that the combination of using BEAGLE and running analyses on powerful graphics cards aimed at the scientific computing market allows for massive performance gains for many challenging data sets. Given that sequence data sets keep growing in size and are being complemented with associated trait data, we have paid particular attention to a popular discrete trait model that parameterizes the transition rates between its states as a GLM, to allow for the inclusion of covariates to help explain transitions in the trait data. Graphics cards can be particularly useful when dealing with such models, as shown in the benchmarks presented, and we have hence presented a number of examples from the literature in which such a setup was used to perform the analyses.

Discrete phylogeography approaches (or discrete trait analyses), as the ones presented in this chapter, treat the sampling locations of the sequences as informative data, rather than uninformative auxiliary variables [9, 23, 25]. As such, the posterior distribution of the parameters given the data contains not only the likelihood of the sequences given the genealogy and substitution model but also the likelihood of the sampling locations given the genealogy and migration matrix, calculated by integrating over all possible discrete state transition histories using Felsenstein's pruning algorithm [16]. What makes this computationally demanding is a potential large number (equal to the number of branches in the phylogeny) of potentially high-dimensional (depending on the number of sampling locations) matrices, which can be parallelized

across a large number of computing cores such as those found on a GPU.

Similarly, structured coalescent approaches also contain the likelihood of the sequences given the genealogy and substitution model in their posterior distribution of the parameters given the data. The use of BEAGLE will yield equal benefits to both approaches when it comes to the computation of the likelihood of the sequences given the genealogy and substitution model. However, rather than the likelihood of the sampling locations, structured coalescent approaches require computation of the probability density of the genealogy and migration history under the structured coalescent given the migration matrix and effective population sizes. To compute this density, a product of exponentials—one for each of the time intervals between successive events (coalescence, sampling, or migration)—needs to be calculated. If the number of demes is sufficiently large, a GPU implementation of the probability density of the genealogy and migration history under the structured coalescent may be able to compute the contribution to this density for each of those time intervals in a highly parallel manner.

Approximations to the structured coalescent include, for example, BASTA [9], which aims to compute the probability density of the genealogy under the structured coalescent, integrated over migration histories. The computational bottleneck of this approach lies with calculating and updating the probability distribution of lineages among demes, over all lineages and over all coalescent events. This involves computing the matrix exponential of the product of each time interval duration with the backwards-in-time migration rate matrix, of which the diagonal elements are defined such that the rows sum to zero. BEAGLE is equipped with a parallel thread block design for computing such finite-time transition probabilities, and to construct the finite-time transition probabilities in parallel across all lineages, and therefore has the potential to provide performance increases for structured coalescent approximations such as BASTA. However, the application software that calls upon BEAGLE needs to be implemented to rely on BEAGLE's API in order to achieve the corresponding performance increases.

Graphics cards aimed at the scientific computing market have traditionally offered roughly three times the single-precision performance compared to their double-precision performance (Tesla K40, K20X and K20). Previous generation cards, such as the Tesla K10, offered poor double-precision performance and focused solely on single-precision performance (up to 24 times their double-precision performance). The latest generation of GPUs, specifically the Tesla P100, offers tremendous double-precision performance, while single-precision performance is still twice as high. We therefore expect a doubling in performance for the

computations described in this chapter if we would run them in single precision on GPU, provided that the decrease in accuracy would not lead to rescaling issues which would slow down the evaluations.

In theory, single-precision likelihood evaluations will be twice as fast as double-precision likelihood evaluations on CPU as well, but with more rescaling issues hampering performance. However, the influence of switching to single precision is more difficult to assess for CPUs, as there are a number of other factors to consider. Single-precision floating points are half the size compared to double-precision floating points and hence they may fit into a lower level of cache with a lower latency, which potentially frees up cache space to cache more (or other) data. Additionally, they require half the memory bandwidth, which frees up that bandwidth for other operations to be performed. Nevertheless, the total overall bandwidth will still be limited compared to that of powerful graphics cards and this will not suffice to bridge the performance differences between CPUs and GPUs in phylogenetics.

Finally, we have presented an interesting avenue for further increasing computational performance on multicore hardware, in the form of a new adaptive MCMC transition kernel. Traditional MCMC transition kernels generally update single parameters in a serial fashion triggering sequential likelihood evaluations on single cores. The adaptive transition kernel however updates a collection of continuous parameters simultaneously, triggering multiple likelihood evaluations in parallel on multiple cores and hence allowing for potentially large improvements in computational efficiency. Further research into this area is needed to continuously advance MCMC kernels and keep computation time manageable for a wide range of models in Bayesian phylogenetics.

7 Notes

1. We have showcased the potentially impressive performance gains brought about by using BEAGLE in conjunction with powerful graphics cards. However, users sometimes complain about the poor performance gains they experience when using a GPU for their analyses, which may have to do with their GPU being not particularly suited for scientific computing. We urge readers to be cautious as to which GPU they invest in, as there is an important distinction between graphics cards aimed at the gaming market and those aimed at the scientific computing market. Computer gaming cards mainly offer tremendous single-precision performance, but typically weak double-precision performance. We hence advise to invest in GPUs aimed at scientific computing, offering increased accuracy and performance in double precision. As a rule, computer gaming

cards have a much reduced cost compared to scientific computing cards, but we advise readers to check the technical specifications of the card before purchase.

2. While 32-bit operating systems are no longer the norm, such systems are still being used from time to time, and problems have been reported in the use and/or installation of BEAGLE on such systems. We strongly advise to install and run BEAST together with BEAGLE on a 64-bit operating system and, in the case of problems, urge users to check that their Java installation is a proper 64-bit installation as well (i.e., avoid 32-bit or mixed mode software installations). The BEAGLE website, hosted at https://github.com/beagle-dev/beagle-lib, contains installation instructions for Windows, Linux/Unix, and Mac systems.
3. While powerful GPUs can be purchased and installed in desktop computers for immediate use, high-performance computing (HPC) centers or computing clusters can also be equipped with GPUs. These systems typically run a job scheduler that allows users to submit BEAST analyses to either CPU or GPU nodes. In case the requested resources are not immediately available, the submitted job is placed in a queue until those resources become available, which may take some time. We hence strongly advise users (especially those who manually compose their input files) to first test their BEAST XML files on a local desktop machine with BEAGLE installed, in order to not have wasted precious time in a job queue only to find out the BEAST XML cannot be run properly.

8 Exercises

1. An important aspect to getting computations—such as those discussed in this chapter—up and running, is defining which hardware is available on your computer or server. This can easily be checked using BEAST once BEAGLE has been installed. To check this when using the BEAST graphical user interface (GUI), simply check the box that says "Show list of available BEAGLE resources and Quit"; alternatively, when using the command-line interface using a BEAST Java Archive (or JAR) file—which can usually be found in the `lib` directory within the BEAST folder—you can simply type:

```
java -jar beast.jar -beagle_info
```

If the path to the BEAGLE library hasn't been set up automatically, be sure to add its location to the command by adding:

```
java -Djava.library.path=/usr/local/lib ...
```

On a typical desktop system equipped with a GPU fit for scientific computing, this will yield the following output to screen:

```
Using BEAGLE library v2.1.2 for accelerated, parallel
likelihood evaluation. 2009-2013, BEAGLE Working Group.
Citation: Ayres et al (2012) Systematic Biology 61: 170-173

BEAGLE resources available:
0 : CPU
    Flags: PRECISION_SINGLE PRECISION_DOUBLE ...

1 : Intel(R) HD Graphics 530
    Global memory (MB): 1536
    Clock speed (Ghz): 1.05
    Number of compute units: 24
    Flags: PRECISION_SINGLE COMPUTATION_SYNCH ...

2 : Tesla K40c
    Global memory (MB): 11520
    Clock speed (Ghz): 0.74
    Number of cores: 2880
    Flags: PRECISION_SINGLE PRECISION_DOUBLE ...
```

In order to determine which resource to use for your computations, it's important to look into the specifications of the GPU as listed by the hardware vendor. For example, certain GPUs will be equipped with a large number of cores and yet they're aimed at the computer gaming market, which will result in poor double-precision performance. As we have shown in Table 1, the Tesla brand is typically well suited for GPU computing, but other cards may be appropriate as well if they deliver adequate double-precision peak performance. In the output printed above, it's quite obvious that we'll be interested in running our analyses on resource 2, i.e., a GPU equipped with thousands of computing cores (resource 1 is an integrated graphics unit, mainly fit for delivering graphics output to screen).

2. Once you have located a GPU fit for scientific computing on your desktop computer or server, try to perform your analysis both on the system's CPU and GPU to compare performance. Using BEAST's GUI, the default option is to run on CPU; if you'd like to run your analysis on a suitable GPU, use BEAST's GUI to select "GPU" where it says "Prefer use of:." However, most desktop computers don't come equipped with powerful

graphics cards, and most often servers aimed at high-performance computing (HPC) will be used for performing these types of computations. As such servers are typically instructed using a command-line interface, BEAST offers the possibility to assign computations to one or more specific GPUs. Using the system described here, it would hence make sense to run your analysis on resource 2, which can be done as follows:

```
java -jar beast.jar -beagle_gpu -beagle_order 2 data.xml
```

Note that not specifying `-beagle_order` will result in the analysis being run on the system's CPU, i.e., resource 0. Additionally, when employing a GPU for your analyses, adding the `-beagle_gpu` argument is highly advised. Many different combinations of using resources arise when your data set is partitioned into multiple subsets, for example if your data is partitioned according to gene and/or codon position. In such cases, it may be beneficial to split those partitions onto multiple resources by using the `-beagle_order` command-line option. For example, the Ebola virus data set (without trait data) has four partitions; it may be useful (although this depends on the actual hardware and needs to be tested) to compute the likelihood of one partition on the CPU (i.e., resource 0) and the other three likelihoods on the GPU (i.e., resource 2). This can be done as follows:

```
java -jar beast.jar -beagle_gpu -beagle_order 0,2,2,2
ebola.xml
```

3. In some cases, such as for example the carnivores data set analyzed in this chapter, only one (sequence) data partition is available. On CPU, drastic performance improvements can still be achieved by using a BEAGLE feature that allows to split up a data partition into multiple subsets, as can be seen in Fig. 5. This approach will lead to performance increases on most CPU systems, as many laptops now come equipped with 4-core processors; this can hence easily be tested on the system you're currently using. To split a (sequence) data partition into two subsets, you can use the following command:

```
java -jar beast.jar -beagle_instances 2 carnivores.xml
```

To generate the results in Fig. 5, we have used this approach to split the data set into 2, 4, 8, 12, 16, 20, and 24 subpartitions, increasing performance every step of the way. Note that on GPU, this approach will only lead to an increased

overhead and hence worse performance compared to keeping the single data partition, as all the likelihood calculations end up on the same (GPU) device. With multiple GPUs in a system however, this can also lead to drastic performance improvements, as shown throughout this chapter.

Acknowledgements

The authors acknowledge support from the European 2020 Union Seventh Framework Programme for research, technological development, and demonstration under Grant Agreement no. 278433-PREDEMICS. The VIROGENESIS project receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 634650. The COMPARE project receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 643476. This work is supported by National Science Foundation (NSF) [DMS 1264153 to M.A.S.; DBI 0755048 and DBI 1356562 to Michael P. Cummings who supported D.L.A. and is also acknowledged here] and National Institutes of Health [R01 AI117011 and R01 HG006139 to M.A.S.]. G.B. acknowledges support from the Interne Fondsen KU Leuven/Internal Funds KU Leuven and of a Research Grant of the Research Foundation—Flanders (FWO; Fonds Wetenschappelijk Onderzoek—Vlaanderen). We acknowledge funding of the "Bijzonder Onderzoeksfonds," KU Leuven (BOF, No. OT/14/115), and the Research Foundation—Flanders (FWO, G0D5117N and G0B9317N). We gratefully acknowledge support from the NVIDIA Corporation with the donation of parallel computing resources used for this research. The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation—Flanders (FWO) and the Flemish Government—Department EWI.

References

1. Ayres DL, Cummings MP (2017) Configuring concurrent computation of phylogenetic partial likelihoods: accelerating analyses using the BEAGLE library. In: 17th International conference on algorithms and architectures for parallel processing: ICA3PP 2017, Collocated Workshops
2. Ayres DL, Cummings MP (2017) Heterogeneous hardware support in BEAGLE, a high-performance computing library for statistical phylogenetics. In: 46th International conference on parallel processing workshops (ICPPW 2017)
3. Ayres DL, Darling A, Zwickl DJ, Beerli P, Holder MT, Lewis PO, Huelsenbeck JP, Ronquist F, Swofford DL, Cummings MP, Rambaut A, Suchard MA (2012) BEAGLE: an application programming interface and high-performance computing library for statistical phylogenetics. Syst Biol 61(1):170–173
4. Baele G, Lemey P (2013) Bayesian evolutionary model testing in the phylogenomics era: matching model complexity with computational efficiency. Bioinformatics 29 (16):1970–1979

5. Baele G, Lemey P, Rambaut A, Suchard MA (2017) Adaptive MCMC in Bayesian phylogenetics: an application to analyzing partitioned data in BEAST. Bioinformatics 33 (12):1798–1805
6. Baele G, Suchard MA, Rambaut A, Lemey P (2017) Emerging concepts of data integration in pathogen phylodynamics. Syst Biol 66(1): e47–e65
7. Bielejec F, Baele G, Vrancken B, Suchard MA, Rambaut A, Lemey P (2016) Spread3: interactive visualization of spatiotemporal history and trait evolutionary processes. Mol Biol Evol 33 (8):2167–2169. https://doi.org/10.1093/molbev/msw082
8. Brockmann D, David V, Gallardo AM (2009) Human mobility and spatial disease dynamics. In: Diffusion fundamentals III. Leipziger Universitätsverlag, Leipzig, pp 55–81
9. De Maio N, Wu CH, O'Reilly KM, Wilson D (2015) New routes to phylogeography: a Bayesian structured coalescent approximation. PLoS Genet 11(8):e1005421
10. Doucet A, De Freitas N, Gordon N (eds) (2001) Sequential Monte Carlo methods in practice. Springer, New York
11. Drummond AJ, Suchard MA, Xie D, Rambaut A (2012) Bayesian phylogenetics with BEAUti and the BEAST 1.7. Mol Biol Evol 29 (8):1969–1973
12. Dudas G, Carvalho LM, Bedford T, Tatem AJ, Baele G, Faria NR, Park DJ, Ladner JT, Arias A, Asogun D, Bielejec F, Caddy SL, Cotten M, D'Ambrozio J, Dellicour S, Caro AD, Diclaro II JD, Durrafour S, Elmore MJ, Fakoli III LS, Faye O, Gilbert ML, Gevao SM, Gire S, Gladden-Young A, Gnirke A, Goba A, Grant DS, Haagmans BL, Hiscox JA, Jah U, Kargbo B, Kugelman JR, Liu D, Lu J, Malboeuf CM, Mate S, Matthews DA, Matranga CB, Meredith LW, Qu J, Quick J, Pas SD, Phan MVT, Pollakis G, Reusken CB, Sanchez-Lockhart M, Schaffner SF, Schieffelin JS, Sealfon RS, Simon-Loriere E, Smits SL, Stoecker K, Thorne L, Tobin EA, Vandi MA, Watson SJ, West K, Whitmer S, Wiley MR, Winnicki SM, Wohl S, Wölfel R, Yozwiak NL, Andersen KG, Blyden SO, Bolay F, Carroll MW, Dahn B, Diallo B, Formenty P, Fraser C, Gao GF, Garry RF, Goodfellow I, Günther S, Happi CT, Holmes EC, Keïta S, Kellam P, Koopmans MPG, Kuhn JH, Loman NJ, Magassouba N, Naidoo D, Nichol ST, Nyenswah T, Palacios G, Pybus OG, Sabeti PC, Sall A, Ströher U, Wurie I, Suchard MA, Lemey P, Rambaut A (2017) Virus genomes reveal factors that spread and sustained the Ebola epidemic. Nature 544(7650):309–315
13. Eisen JA, Fraser CM (2003) Phylogenomics: intersection of evolution and genomics. Science 300(5626):1706–1707
14. Faria NR, Suchard MA, Rambaut A, Streicker DG, Lemey P (2013) Simultaneously reconstructing viral cross-species transmission history and identifying the underlying constraints. Philos Trans R Soc B 368 (1614):20120196
15. Felsenstein J (1978) The number of evolutionary trees. Syst Biol 27(1):27–33
16. Felsenstein J (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol 17(6):368–376
17. Gilks WR, Richardson S, Spiegelhalter DJ (1996) Markov chain Monte Carlo in practice. Chapman and Hall, London
18. Goldman N, Yang Z (1994) A codon-based model of nucleotide substitution for protein-coding DNA sequences. Mol Biol Evol 11 (5):725–736. http://mbe.oxfordjournals.org/content/11/5/725.abstract, http://mbe.oxfordjournals.org/content/11/5/725.full.pdf+html
19. Guindon S, Dufayard JF, Lefort V, Anisimova M, Hordijk W, Gascuel O (2010) New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. Syst Biol 59 (3):307–321
20. Huelsenbeck JP, Ronquist F, Nielsen R, Bollback JP (2001) Bayesian inference of phylogeny and its impact on evolutionary biology. Science 294(5550):2310–2314
21. Jeffroy O, Brinkmann H, Delsuc F, Philippe H (2006) Phylogenomics: the beginning of incongruence? Trends Genet 22(4):225–231
22. Kobert K, Flouri T, Aberer A, Stamatakis A (2014) The divisible load balance problem and its application to phylogenetic inference. In: International workshop on algorithms in bioinformatics, pp 204–216
23. Kumar S, Filipski AJ, Battistuzzi FU, Pond SLK, Tamura K (2012) Statistics and truth in phylogenomics. Mol Biol Evol 29(2):457–472
24. Lemey P, Rambaut A, Drummond AJ, Suchard MA (2009) Bayesian phylogeography finding its roots. PLoS Comput Biol 5(9):e1000520
25. Lemey P, Rambaut A, Bedford T, Faria N, Bielejec F, Baele G, Russell CA, Smith DJ, Pybus OG, Brockmann D, Suchard MA (2014) Unifying viral genetics and human transportation data to predict the global transmission dynamics of human influenza H3N2. PLoS Pathog 10(2):e1003932
26. Muse SV, Gaut BS (1994) A likelihood approach for comparing synonymous and

nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. Mol Biol Evol 11(5):715–724

27. Neal RM (2010) MCMC using Hamiltonian dynamics. In: Handbook of Markov chain Monte Carlo, vol 54. CRC Press, Boca Raton, pp 113–162
28. Roberts GO, Rosenthal JS (2009) Examples of adaptive MCMC. J Comput Graph Stat 18:349–367
29. Ronquist F, Teslenko M, van der Mark P, Ayres DL, Darling A, Höhna S, Larget B, Liu L, Suchard MA, Huelsenbeck JP (2012) MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. Syst Biol 61(3):539–542
30. Sanmartín I, van der Mark P, Ronquist F (2008) Inferring dispersal: a Bayesian approach to phylogeny-based island biogeography, with special reference to the Canary Islands. J Biogeogr 35:428–449
31. Streicker DG, Turmelle AS, Vonhof MJ, Kuzmin IV, McCracken GF, Rupprecht CE (2010) Host phylogeny constrains cross-species emergence and establishment of rabies virus in bats. Science 329(5992):676–679
32. Suchard MA, Rambaut A (2009) Many-core algorithms for statistical phylogenetics. Bioinformatics 25:1370–1376
33. Yang Z (1996) Among-site rate variation and its impact on phylogenetic analyses. Trends Ecol Evol 11(9):367–372
34. Yang Z, Rannala B (1997) Bayesian phylogenetic inference using DNA sequences: a Markov chain Monte Carlo method. Mol Biol Evol 14(7):717–724
35. Zwickl DJ (2006) Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. PhD thesis, The University of Texas at Austin