Exploiting Multi-grain Parallelism for Efficient Selective Sweep Detection

Nikolaos Alachiotis, Pavlos Pavlidis, and Alexandros Stamatakis

The Exelixis Lab, Scientific Computing Group
Heidelberg Institute for Theoretical Studies
Heidelberg, Germany
{Nikolaos.Alachiotis,Pavlos.Pavlidis,Alexandros.Stamatakis}@h-its.org

Abstract. Selective sweep detection localizes targets of recent and strong positive selection by analyzing single nucleotide polymorphisms (SNPs) in intra-species multiple sequence alignments. Substantial advances in wet-lab sequencing technologies currently allow for generating unprecedented amounts of molecular data. The increasing number of sequences and number of SNPs in such large multiple sequence alignments cause prohibiting long execution times for population genetics data analyses that rely on selective sweep theory. To alleviate this problem, we have recently implemented fine- and coarse-grain parallel versions of our open-source tool OmegaPlus for selective sweep detection that is based on the ω statistic. A performance issue with the coarse-grain parallelization is that individual coarse-grain tasks exhibit significant run-time differences, and hence cause load imbalance. Here, we introduce a significantly improved multi-grain parallelization scheme which outperforms both the fine-grain as well as the coarse-grain versions of OmegaPlus with respect to parallel efficiency. The multi-grain approach exploits both coarse-grain and fine-grain operations by using available threads/cores that have completed their coarse-grain tasks to accelerate the slowest task by means of fine-grain parallelism. A performance assessment on real-world and simulated datasets showed that the multi-grain version is up to 39% and 64.4% faster than the coarse-grain and the fine-grain versions, respectively, when the same number of threads is used.

1 Introduction

Charles Darwin attributed evolution to natural selection among species. Natural selection occurs when members of a population of species die and are replaced by offsprings that are better adapted to survive and reproduce in a given environment. The field of population genetics studies the genetic composition of populations as well as changes in this genetic composition that are, for instance, driven by natural selection or genetic drift. The input data for population genetic analyses is a multiple sequence alignment (MSA), essentially an $n \times m$ data matrix that contains n DNA sequences with a length of m nucleotide characters each (also denoted as columns or alignment sites). Recent advances in sequencing technology (e.g., Next-Generation Sequencers (NGS)) are currently generating a

Y. Xiang et al. (Eds.): ICA3PP 2012, Part I, LNCS 7439, pp. 56–68, 2012.

spectacular amount of molecular sequence data because entire genomes can now be rapidly and accurately sequenced at low cost. Therefore, we urgently need to adapt, optimize, and parallelize state-of-the-art tools for population genetic data analysis such that they scale to large emerging genomic datasets and keep pace with the molecular data avalanche.

In population genetics, statistical tests that rely on selective sweep theory [1] are widely used to identify targets of recent and strong positive selection. This is achieved by analyzing single nucleotide polymorphisms (SNPs) in intra-species MSAs. A recently introduced method for detecting selective sweeps is the ω statistic [2], which has been implemented by Jensen et al. [3] and Pavlidis et al. [4]. We recently released OmegaPlus (`http://www.exelixis-lab.org/software.html`), a fast sequential open-source implementation of the ω statistic. OmegaPlus has lower memory requirements than competing codes and can analyze whole-genome MSAs on off-the-shelf multi-core desktop systems. The kernel function of OmegaPlus that dominates execution times consist of a dynamic programming algorithm for calculating sums of so-called linkage-disequilibrium (LD) values in sub-genomic regions (fractions of the genome). We also developed two parallel versions: OmegaPlus-F, which parallelizes the dynamic programming matrix computations at a fine-grain level, and OmegaPlus-C, which divides the alignment into sub-regions and assigns these individual and independent tasks to threads (coarse-grain parallelism).

The above, relatively straightforward parallelization schemes yielded acceptable, yet sub-optimal speedups because both approaches exhibit some drawbacks. Exploiting fine-grain parallelism in dynamic programming matrices, especially when these are relatively small, can exhibit unfavorable computation-to-synchronization ratios. This is the case in OmegaPlus-F, despite the fact that we used an efficient thread synchronization strategy based on busy waiting [5]. Thus, because of the relatively small amount of operations required per dynamic programming matrix cell, the fine-grain parallel version only scales well if the input MSA comprises hundreds to thousands of sequences. Unfortunately, the coarse-grain OmegaPlus-C version also faces a parallel scalability problem: the variability of the density of SNPs in the MSA can cause substantial load imbalance among threads. Despite the fact that all sub-genomic regions are of the same size, tasks that encompass regions with high SNP density require significantly longer execution times. Note that the time required to compute the ω statistic in a sub-genomic region (a task) increases quadratically with the number of SNPs in that region. Hence, the parallel performance of OmegaPlus-C is limited by the slowest thread/task, that is, the thread assigned to the sub-genomic region with the highest SNP density.

To alleviate this load imbalance issue, we developed a multi-grain parallelization strategy that combines the coarse- and fine-grain approaches. The underlying idea is to use fast threads that have completed their task to accelerate the ω statistic computations of slower threads in a fine-grain way. In other words, when a thread completes its coarse-grain task early, it is assigned to help the thread that is expected to finish last at that particular point in time. Multi-grain

parallelization approaches have also been reported in evolutionary placement of short reads [6] as well as to compute the phylogenetic likelihood function on the Cell Broadband Engine [7].

The remainder of this paper is organized as follows: In Section 2, we describe the ω statistic that relies on linkage disequilibria, and we briefly outline the fine- and coarse-grain parallelization approaches in Section 3. Thereafter, we introduce the multi-grain approach and the implementation of the dynamic helper thread assignment mechanism (Section 4). A detailed performance evaluation of the multi-grain version of OmegaPlus is provided in Section 5. We conclude in Section 6 and address directions of future work.

2 The LD Pattern of Selective Sweeps

The LD is used to capture the non-random association of states at different MSA positions. Selective sweep theory [1] predicts a pattern of excessive LD in each of the two alignment regions that flank an evolutionarily advantageous *and* recently fixed mutation. This genomic pattern can be detected by using the ω statistic.

Assume a genomic window with S SNPs that is split into a left and a right sub-region with l and $S-l$ SNPs, respectively. The ω statistic is then computed as follows:

$$\omega = \frac{\left(\binom{l}{2} + \binom{S-l}{2}\right)^{-1}\left(\sum_{i,j\in L} r_{ij}^2 + \sum_{i,j\in R} r_{ij}^2\right)}{(l(S-l))^{-1}\sum_{i\in L, j\in R} r_{ij}^2}, \tag{1}$$

where r_{ij}^2 represents one common LD measure that simply is the squared correlation coefficient between sites i and j. The ω statistic quantifies to which extent average LD is increased on either side of the selective sweep (see numerator of Equation 1) but *not across* the selected site (see denominator of Equation 1). The area between the left and right sub-regions is considered as the center of the selective sweep.

In sub-genomic regions, that is, candidate regions of limited length (some thousand bases/sites long), the ω statistic can be computed at each interval between two SNPs. S refers to the total number of SNPs, and the goal is to detect that l which will maximize the ω statistic. When scanning whole-genome datasets, the analysis becomes more complicated. Evaluating the ω statistic at each interval between two SNPs can become computationally prohibitive since hundreds of thousands of SNPs may occur in an entire chromosome. Furthermore, S can not be defined as the overall amount of SNPs along the whole chromosome. This would also be biologically meaningless because a selective sweep usually only affects the polymorphic patterns in the neighborhood of a beneficial (advantageous) mutation.

To process whole-genome datasets, we assume a grid of equidistant locations L_i, $1 < i < k$; k is defined by the user, for which the ω statistic is computed. We also assume a user-defined sub-genomic region of size R_{MAX} that extends to either side of a beneficial mutation. Such a sub-genomic region represents the

genomic area (neighborhood) in which polymorphic patterns may have been affected by the selective sweep. OmegaPlus evaluates the ω statistic for all possible sub-regions that are enclosed in R_{MAX} and reports the maximum ω value as well as the sub-region size that maximizes the ω statistic. Figure 1 illustrates two consecutive steps of the ω statistic calculation at a specific location. The user-defined R_{MAX} value determines the left and right borders (vertical thin lines). The ω statistic is computed for all possible sub-regions that lie within the left and right borders, and the maximum ω value is reported. The process is repeated for all locations for which the ω statistic needs to be computed.

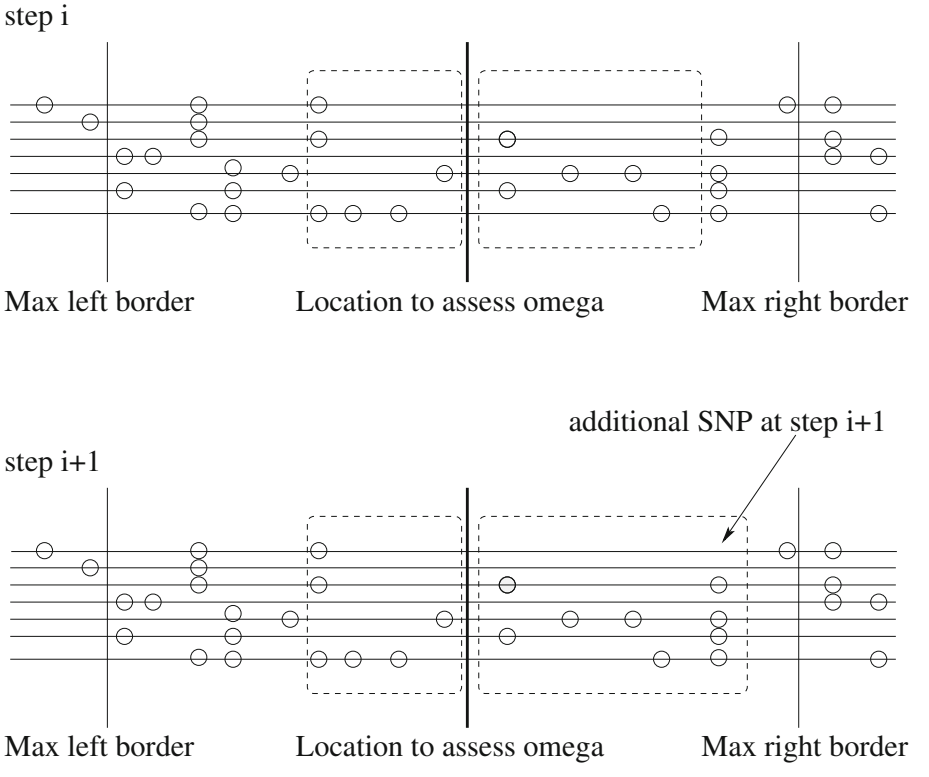

Fig. 1. The process of detecting the sub-regions that maximize the ω statistic for a given location/selective sweep. The goal is to evaluate the ω statistic at the alignment position denoted by the thick vertical line. The thin vertical lines to the left and right side of the thick line indicate the sub-region borders as defined by R_{MAX}. At step i, only the SNPs enclosed within the dashed-line areas contribute to the calculation of the ω statistic. At step $i + 1$, one more SNP that belongs to the right sub-region contributes to the ω statistic calculation. Calculations are repeated for all possible sub-regions within the left and right borders (vertical thin lines). The maximum ω value and the associated sub-region sizes are then reported.

An overview of the computational workflow of OmegaPlus on whole-genome datasets is provided in Figure 2. The Figure shows the basic steps for the calculation of an ω statistic value at a given position. The same procedure is repeated to compute ω statistic values at all positions that have been specified by the user. Initially, we check whether sub-genomic regions, as defined by R_{MAX}, overlap. If they overlap, the data is re-indexed to avoid recalculation of previously already computed values and thereby save operations. Thereafter, all required new LD values are computed as well as all sums of LDs in the region. Finally, all possible ω statistic values for the specific region are computed.

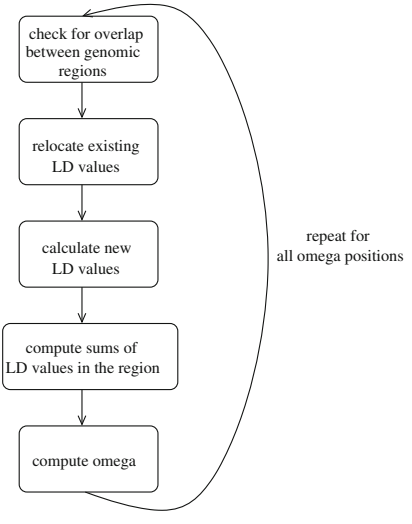

Fig. 2. The basic algorithmic steps of OmegaPlus for the computation of the ω statistic at a number of alignment positions requested by the user

An algorithmic advance of the ω statistic implementation in OmegaPlus over previous approaches is that it deploys a dynamic programming algorithm to calculate all $\sum_{i \in L, j \in R} r_{ij}^2$ values in a sub-genomic region G. Given S SNPs in G, a matrix M of size $S^2/2$ is computed as follows:

$$M_{i,j} = \begin{cases} 0 & 1 \leq i \leq S,\ j = i \\ r_{ij} & 2 \leq i \leq S,\ j = i-1 \\ M_{i,j+1} + M_{i-1,j} + & \\ M_{i-1,j+1} + r_{ij} & 3 \leq i < S,\ i-1 > j \geq 0. \end{cases} \tag{2}$$

Thereafter, all $\sum_{i \in L, j \in R} r_{ij}^2$, $\sum_{i,j \in L} r_{ij}^2$, and $\sum_{i,j \in R} r_{ij}^2$ values required by Equation 1 are retrieved from the matrix.

3 Fine- and Coarse-Grain Parallelizations

The fine-grain parallelization of Equation 2 is implemented by evenly distributing the independent r_{ij}^2 calculations (that dominate run-times for computing M) among all threads. Furthermore, the computation of all ω values at an alignment position (one ω value for every l in Equation 1) is also evenly distributed among threads. Figure 3 illustrates how the basic algorithmic steps of Figure 2 are executed by multiple threads using this fine-grain parallelization scheme. As already mentioned in Section 1, the computation of the r_{ij}^2 values is relatively fast for a small number of sequences compared to thread synchronization times. Furthermore, if the infinite site model [8] is assumed, the DNA input data can be transformed into a binary data representation. In this case, r_{ij}^2 computations become even faster because computing r_{ij}^2 on DNA data requires approximately 10 times (on average) more arithmetic operations than on binary data.

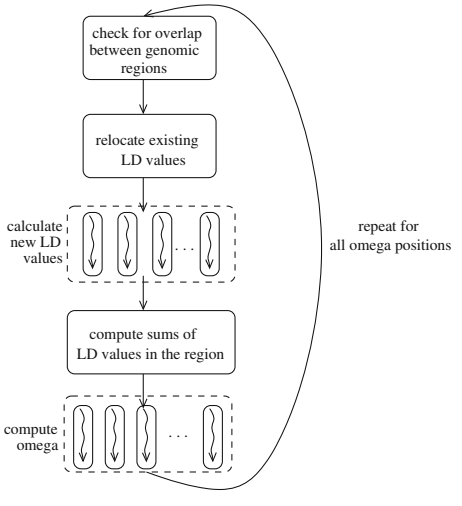

Fig. 3. Fine-grain OmegaPlus parallelization

Unlike the fine-grain approach, our coarse-grain scheme is not affected by an unfavorable computation-to-synchronization ratio since no thread synchronization is required. Each thread is assigned a different sub-genomic region (part of the input MSA) and carries out all ω statistic operations in that region. Note that we generate as many sub-genomic regions (tasks) as there are threads/cores available. Thus, threads need to synchronize only once to determine if they have all completed the analysis of their individual sub-genomic region/task. Figure 4 outlines this coarse-grain parallelization scheme. Since synchronization is only required to ensure that all tasks have been completed, the performance of the coarse-grain approach is limited by the run time of the slowest thread/task. The slowest thread is always the one that has been assigned to the sub-genomic region with the highest number of SNPs.

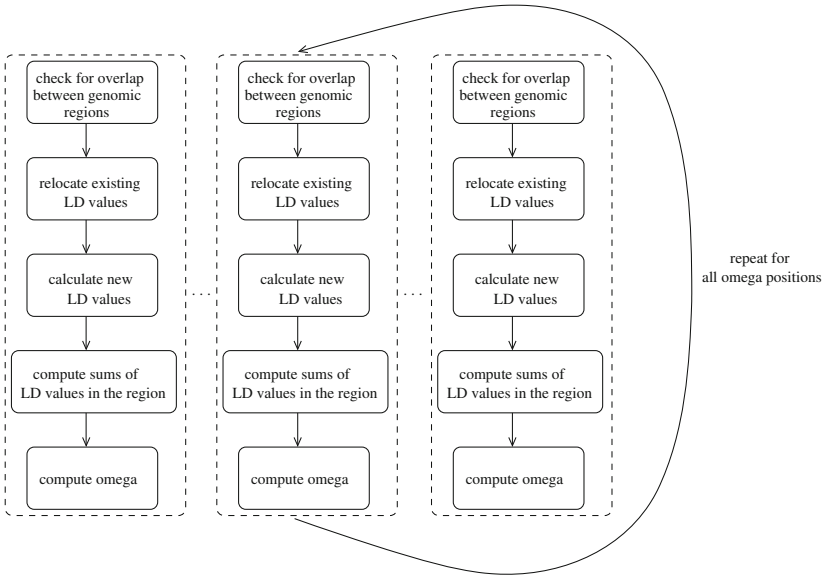

Fig. 4. Coarse-grain OmegaPlus parallelization

4 Multi-grain Parallelization

To alleviate the load imbalance caused by regions of high SNP density, we introduce a multi-grain parallelization scheme for reducing the runtime of the slowest threads/tasks by using a hybrid coarse-grain/fine-grain strategy.

4.1 Underlying Idea

During the first stages, the approach is similar to the coarse-grain scheme, since each thread is assigned a sub-genomic region of equal size (but potentially distinct density) and carries out the ω statistic operations independently and sequentially within that region. However, when a thread finishes processing its task/region in the coarse-grain approach, it starts executing a busy-wait until the last thread has completed its task. As already mentioned, the threads are only synchronized once via such a busy-wait barrier in the very end. In the multi-grain implementation, when a thread finishes processing its own task, it notifies the other threads that are still working on their tasks that it is available to help accelerate computations. At each point in time, only the slowest among all threads will obtain help from all other available threads that have completed their tasks. Henceforth, we use the term *temporary workers* to refer to the threads that are available to help others and the term *temporary master* to refer to the thread that is getting help from the *temporary workers*.

The *temporary master* is responsible to assign temporary thread IDs to the *temporary workers*, synchronize them, and release them again after a certain

period of time. The *temporary master* can not use the *temporary workers* for an unlimited amount of time since another thread will become the slowest thread at some later point in time because of the speedup in computations attained by using the *temporary workers*. The current *temporary master* is entitled to announce the next *temporary master*. This approach is used to prevent slow threads to compete for *temporary workers* and also to avoid excessive occupation of the *temporary workers* by the same *temporary master*. Every *temporary master* occupies the workers for the calculation of the ω statistic at 25 positions. This number has been determined experimentally. Smaller numbers lead to increased synchronization overhead between threads while larger numbers lead to larger execution time differences between the slowest threads. While the *temporary master* is working with the *temporary workers* using fine-grain parallelism, the other threads (on remaining regions) are still operating in coarse-grain mode. When a *temporary master* renounces its privilege to use the *temporary workers* it returns to coarse-grain parallel mode. Figure 5 outlines the possible operational states of a thread during the analysis of a MSA using the multi-grain approach.

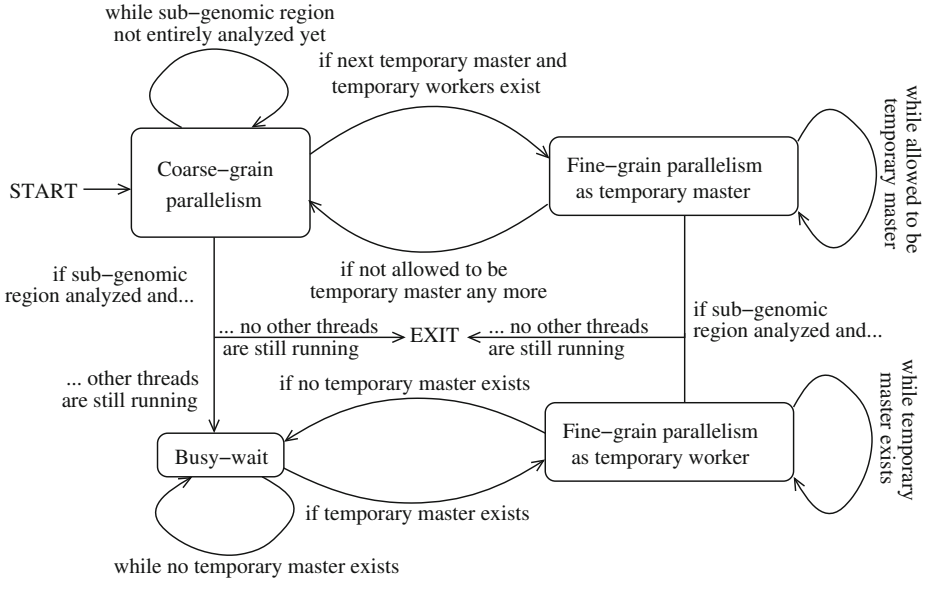

Fig. 5. Processing states of a thread according to the multi-grain parallel model in OmegaPlus

4.2 Implementation

The multi-grain parallelization and synchronization is implemented via integer arrays (*available* array, *turn* array, *progress* array). The *available* array shows which threads have completed their task and are ready to help others with computations. The *turn* array indicates which thread is allowed to occupy the *temporary workers* at each point in time. Finally, the *progress* array shows the progress

each thread has made with the analysis of its sub-genomic region. Progress is measured as the number of calculated L_i values ($1 < i < k$, k is defined by the user, see Section 2) that lie in the sub-genomic region of the thread.

At a given point in time, the slowest thread is determined by searching for the highest number of—still uncalculated—L_i values in the *progress* array. The *temporary master* scans the *progress* array and announces which thread shall become the next *temporary master* by updating the *turn* array. After each L_i value computation by every unfinished thread, all threads check their corresponding entry in the *turn* array to find out whether it is their turn to utilize the *temporary workers*. The thread that is allowed to utilize the *temporary workers* acquires them by marking them as unavailable in the *available* array. Then, temporary thread IDs are assigned by the *temporary master* to all available workers in the *worker_ID* array. The *temporary master* also assigns a temporary thread ID to itself. The temporary thread ID assignment guarantees correct indexing and work distribution in fine-grain parallel computations of the M matrix. For the *temporary master*, the temporary ID also corresponds to the total number of parallel threads that will be used to accelerate computations. The *temporary master* also informs the *temporary workers* who the *temporary master* is via the *master_ID* array. It is important for the *temporary workers* to know the ID of their *temporary master* such as to avoid storing L_i values that correspond to the sub-genomic region of the *temporary master* in memory space that has been allocated by other threads that are still running.

5 Performance Evaluation

The multi-grain parallelization approach is available in the OmegaPlus software (available under GNU GPL at `http://www.exelixis-lab.org/software.html`). Compiling the code with the appropriate `Makefile` will generate the OmegaPlus-M executable (M stands for Multi). To evaluate performance of the multi-grain parallelization, we run experiments using both real-world and simulated datasets and compared execution times among all three parallelization schemes.

To generate simulated data, we used the ms tool by Hudson [9]. We assumed past population size changes (population bottleneck [10]) and positive recombination probability between base pairs. Simulating population bottlenecks in conjunction with recombination events increases the variance between genomic regions [11], thus leading to higher variability in SNP density. The simulated alignment was generated with the following command:

```
ms 1000 1 -r 50000 1000 -eN 0.001 0.001 -eN 0.002 10 -s 10000.
```

Flags `-eN 0.001 0.001` and `-eN 0.002 10` specify a population bottleneck. Backward in time (from present to past), the population has contracted to the 0.001 of its present-day size at time 0.001. Then, at time 0.002, the population size became 10 times larger than the present-day population size. Time is measured in $4N$ generations [9], where N is the effective population size. Regarding recombination (`-r 50000 1000`), we assume that there are 1,000 potential

breakpoints where recombination may occur and the total rate of recombination is 50,000 (i.e. $4Nr = 50000$, where r is the recombination rate between two adjacent sites in the alignment and N the effective population size). Furthermore, we analyzed the X chromosome of 37 *Drosophila melanogaster* genomes sampled in Raleigh, North Carolina. The sequences (available from the Drosophila Population Genomics Project at `http://www.dpgp.org`) were converted from fastq to fasta format using a conversion script (also available at `http://www.dpgp.org`). We obtained a DNA alignment comprising 22,422,827 sites with 339,710 SNPs.

As test platform we used an AMD Opteron 6174 12-core Magny-Cours processor running at 2.2 GHz. Table 1 provides the total execution times required by OmegaPlus-F (fine-grain), OmegaPlus-C (coarse-grain), and OmegaPlus-M (multi-grain) using 2 up to 12 cores for analyzing an average-size alignment with 1,000 sequences and 100,000 alignment sites comprising 10,000 SNPs. All OmegaPlus runs calculated the ω statistic at 10,000 positions along the alignment assuming that the maximum size of the neighborhood of a beneficial mutation that a selective sweep might affect is 20,000 alignment sites. The sequential version of the code (OmegaPlus) required 317 seconds to process this alignment on one core of the test platform.

Table 1. Total execution times (in seconds) of OmegaPlus-F, OmegaPlus-C, and OmegaPlus-M to analyze an alignment of 1,000 sequences and 100,000 sites (10,000 SNPs). The last two rows show the performance improvement of the multi-grain version over the fine-grain (Multi vs Fine) and the coarse-grain (Multi vs Coarse) implementations, respectively.

Parallelization approach	Number of threads					
	2	4	6	8	10	12
Fine-grain	211.1	156,6	126,0	113.3	107.5	99.5
Coarse-grain	237.8	214,4	164,5	110,7	116.1	101.1
Multi-grain	197.8	135,0	100,3	87,2	77.1	63.8
Multi vs Fine (%)	6.3	13.7	20.3	23.0	28.2	35.9
Multi vs Coarse (%)	16.8	37.0	39.0	21.2	33.6	36.9

The last two rows of Table 1 show the performance improvement over the fine- and coarse-grain parallel implementations for different numbers of threads. On this simulated dataset, the multi-grain approach is between 6.3% (using two threads) and 35.9% (using 12 threads) faster than the fine-grain approach. In comparison to the coarse-grain parallelization, the multi-grain approach is between 16.8% (using two threads) and 39.0% (using 6 threads) faster.

The improvement differences (last two rows of Table 1) among runs with different number of threads are caused by changes in the size of the sub-genomic regions when coarse-grain parallelization is used. Since there are as many sub-genomic regions as threads available, the number of threads affects the size of these regions and as a consequence the variability of SNP density among them. The improving parallel efficiency of OmegaPlus-M with increasing number of

threads when compared to OmegaPlus-F can be attributed to the effectiveness of the initial coarse-grain parallel operations on binary data. As already mentioned, ω statistic computations on binary data require 10 times less operations (on average) than on DNA data. Therefore, when the fine-grain parallel scheme is used to analyze binary data, the synchronization-to-computation ratio is high in comparison to a DNA data analysis. As a consequence, the coarse-grain parallelization is more efficient to analyze binary data than the fine-grain approach.

Table 2 shows the results of the runs on the real-world *Drosophila melanogaster* MSA. Once again, we calculated the ω statistic at 10,000 positions along the alignment, but this time using a maximum neighborhood size of 200,000 alignment sites. On a single core of our test platform, the sequential version required 1,599.8 seconds. Once again, OmegaPlus-M outperformed OmegaPlus-F and OmegaPlus-C in all runs with different number of threads. OmegaPlus-M yielded parallel performance improvements ranging between 22.4% and 64.4% over OmegaPlus-F and between 1.0% and 17.7% over OmegaPlus-C.

Unlike the results on the simulated dataset, the performance continuously improves on the real dataset as we use more threads. This is because the actual number of SNPs in the *Drosophila melanogaster* alignment is one order of magnitude larger than the number of SNPs in the simulated dataset. Despite the fact that the number of ω statistic positions per region is the same in both MSAs, the ω statistic computations on the real-world dataset comprise approximately 45,000 to 60,000 SNPs in every region whereas only 2,500 to 6,000 are present in the simulated dataset. A detailed analysis of the computational load per region revealed that the size of the dynamic programming matrices when the *Drosophila melanogaster* DNA MSA (average number of cells: 4,859,088) is analyzed is twice the size of the respective matrices for the analysis of the simulated binary dataset (average number of cells: 2,239,508). The load analysis also revealed that the overlap between neighborhoods around consecutive ω statistic positions is higher for the simulated dataset. As a consequence, a significantly higher amount of operations is required per matrix for the *Drosophila melanogaster* MSA thus reducing the synchronization-to-computation ratio in the fine-grain scheme and allowing better scalability. Therefore, the more pronounced performance advantages are obtained by deploying fine-grain computations in conjunction with coarse-grain computations in OmegaPlus-M as we increase the number of threads.

The multi-grain approach aims at improving the parallel efficiency of the coarse-grain approach. This is achieved by reducing the runtime of the slowest threads/tasks by means of fine-grain parallelism. Figure 6 depicts the effect of the multi-grain parallelization on the execution times of the individual threads/tasks for the case of using 12 threads to analyze the real-world *Drosophila melanogaster* MSA. The red and blue horizontal lines show the points in time when the slowest threads in OmegaPlus-C and OmegaPlus-M terminate, respectively.

Table 2. Total execution times (in seconds) of OmegaPlus-F, OmegaPlus-C, and OmegaPlus-M to analyze a real-world dataset that comprises 37 sequences and 22,422,827 alignment sites (339,710 SNPs).

Parallelization approach	Number of threads					
	2	4	6	8	10	12
Fine-grain	1105.8	760.8	599.9	565.6	555.1	534,0
Coarse-grain	866.5	477.8	348.9	277.3	247.2	231.7
Multi-grain	857.1	437.3	312.1	248.0	210.8	190.6
Multi vs Fine (%)	22.4	42.5	47.9	56.1	62.1	64.4
Multi vs Coarse (%)	1.0	8.3	10.3	10.4	14.9	17.7

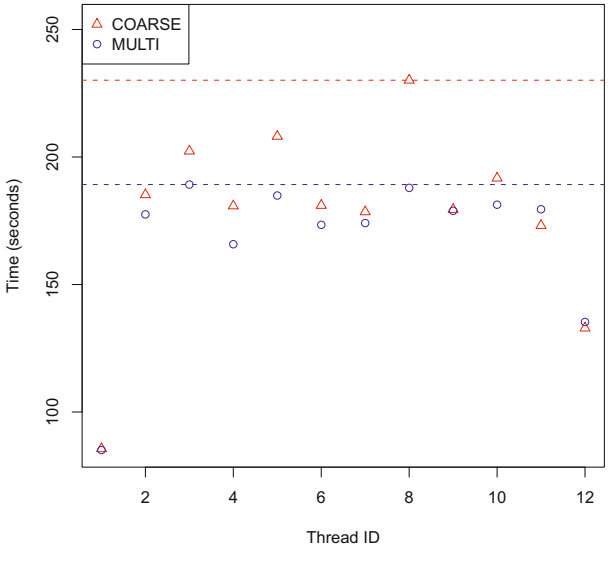

Fig. 6. Per-thread execution times (in seconds) of OmegaPlus-C and OmegaPlus-M when the *Drosophila melanogaster* MSA is analyzed

6 Conclusion and Future Work

We described, implemented, evaluated, and made available an efficient multi-grain parallelization scheme in the open-source OmegaPlus population genetics code. Fast threads that finish their own tasks early can subsequently help slower threads in accomplishing their tasks by deploying fine-grain parallelism. The multi-grain parallelization scheme, in conjunction with previous algorithmic and data structure optimizations in OmegaPlus, now allows for seamless computation of the ω statistic on large whole-genome datasets using off-the-shelf multi-core desktop and server systems.

In terms of future work, we intend to explore alternative strategies for scheduling/assigning fast threads to slower threads that can potentially further increase parallel efficiency. At present, we assign *all* available threads to help only one thread, that is, the thread that will most likely terminate last at each point in time. We expect that, a pre-analysis of MSA SNP density in combination with a detailed empirical investigation of thread behavior as a function of the assigned alignment region will allow for further optimizing *worker threads* to *master thread(s)* assignments. In addition, we plan to explore alternative hardware platforms such as reconfigurable devices and GPUs for offloading the compute-intensive parts of OmegaPlus.

References

1. Maynard Smith, J., Haigh, J.: The hitch-hiking effect of a favourable gene. Genet. Res. 23(1), 23–35 (1974)
2. Kim, Y., Nielsen, R.: Linkage disequilibrium as a signature of selective sweeps. Genetics 167(3), 1513–1524 (2004)
3. Jensen, J.D., Thornton, K.R., Bustamante, C.D., Aquadro, C.F.: On the utility of linkage disequilibrium as a statistic for identifying targets of positive selection in nonequilibrium populations. Genetics 176(4), 2371–2379 (2007)
4. Pavlidis, P., Jensen, J.D., Stephan, W.: Searching for footprints of positive selection in whole-genome snp data from nonequilibrium populations. Genetics 185(3), 907–922 (2010)
5. Berger, S.A., Stamatakis, A.: Assessment of barrier implementations for fine-grain parallel regions on current multi-core architectures. In: Proc. IEEE Int Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS) Conf., pp. 1–8 (2010)
6. Stamatakis, A., Komornik, Z., Berger, S.A.: Evolutionary placement of short sequence reads on multi-core architectures. In: Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2010), pp. 1–8. IEEE Computer Society Press, Washington (2010)
7. Blagojevic, F., Nikolopoulos, D.S., Stamatakis, A., Antonopoulos, C.D.: Dynamic multigrain parallelization on the cell broadband engine. In: Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. PPoPP 2007, pp. 90–100. ACM, New York (2007)
8. Kimura, M.: The number of heterozygous nucleotide sites maintained in a nite population due to steady ux of mutations. Genetics 61(4), 893–903 (1969)
9. Hudson, R.R.: Generating samples under a wright-fisher neutral model of genetic variation. Bioinformatics 18(2), 337–338 (2002)
10. Gillespie, J.H.: Population genetics: a concise guide. Johns Hopkins Univ. Pr. (2004)
11. Haddrill, P.R., Thornton, K.R., Charlesworth, B., Andolfatto, P.: Multilocus patterns of nucleotide variability and the demographic and selection history of drosophila melanogaster populations. Genome Res. 15(6), 790–799 (2005)