Accelerating Single-Step Evaluations Through GPU Offloading

A. Freudenberg[1], M. Schlather[1], J. Vandenplas[2] and T. Pook[2]

[1] Institute of Mathematics, University of Mannheim, B 6, 26 Mannheim, 68159, Germany

[2] Animal Breeding and Genomics, Wageningen UR, P.O. 338, 6700 AH Wageningen, The Netherlands

Abstract

Single Nucleotide Polymorphism (SNP) genotype datasets used in empirical research are steadily growing in size which has introduced challenges in the calculation of population statistics that are based on large parts of the genome. In other fields, similar computational challenges have been tackled with the help of Graphics Processing Units (GPUs). We have developed a range of algorithms for the calculation of SNP genotype matrix operations widely used in empirical studies, which take advantage of modern NVIDIA GPUs. We provide an implementation in the C library miraculix and exemplary interfaces in Julia and Fortran. To ease adaptation, we also supply functions to calculate a number of derivatives, such as the genomic relationship matrix (GRM), linkage disequilibrium (LD) statistics, the genomic BLUP, and principal components analysis. Source code is released under the Apache 2.0 license and is freely available at GitHub. The library is developed in C, C++ and CUDA.

Key words: Genomic prediction, single-step models, high-performance computing, GPU

Introduction

Due to the emergence of high-throughput sequencing technology, the recent decades have seen the collection of massive genomic datasets, furthering the research in various fields in genetics such as human medicine or animal breeding and plant breeding. The consideration of large amounts of data helps to increase the accuracy of predictive models (Canela-Xandri et al., 2016; Zhao et al.,2021; Singh and Prasad, 2021) and some authors show that big data can contribute towards the closing of the missing heritability gap (Kim et al., 2017; Pallares, 2019). For breeding purposes, the use of genomic information leads to more accurate breeding values at earlier life stages, thus allowing for earlier selection to both reduce housing cost and increase genetic gain (Schaeffer, 2006). However, the computational analysis of these datasets places a significant burden on researchers and practitioners.

A genomic relationship matrix (GRM) describes the proportion of the genome that is shared between individuals in a population (Mrode, 2014) and is used in various selection methods such as genomic BLUP (VanRaden, 2008), single-step genomic BLUP (Misztal et al., 2009), extended genomic BLUP for modeling epistatic effects (Jiang and Reif, 2015) or (selective) epistatic random regression BLUP (Vojgani et al., 2021). Similarly, linkage disequilibrium (LD) measures the statistical similarity of pairs of SNPs in a population. For instance, LD quantities are used in human genetic studies to infer information on disease causes or population history (Pritchard and Przeworski, 2001; Gazal et al., 2017). Due to the large dimensions of modern genomic data sets, a naive calculation of the GRM, LD and their derivatives would inflict extraordinarily high computational demands, both in terms of memory requirements and calculation times. Since the SNP genotype of an individual is coded as 0 for one homozygous genotype, 1 for the heterozygous genotype, or 2 for the alternate homozygous genotype, each SNP value can be stored in 2 bits of memory. An example of this compressed storage format is the PLINK 1 binary format (Chang et al., 2015). While many statistical quantities in genomics can be calculated using highly optimized BLAS libraries, similar utilities are not available for these compressed storage formats. There exist two main approaches to mitigate this problem. The first one decompresses SNP genotype data before further processing. For instance, the R packages AGHmatrix (Amadeu et al., 2016), qgg (Rohde et al., 2019), rrBLUP (Endelman, 2011) and snpReady (Granato et al., 2018) use custom floating-point matrix operations for the calculation of the GRM or rely on BLAS libraries. The R package SNPRelate (Zheng et al., 2012) benefits from explicit SIMD instructions in the calculation of LD and the GRM. Standalone solutions for the calculation of LD include HaploView and LDkit (Barrett et al., 2004; Yao, 2020). The calculation of the GRM and LD statistics is also implemented in the software packages PLINK and GCTA (Yang et al., 2011; Chang et al., 2015) which have popularized the second approach for processing compressed genotype data. They both utilize bit-compressed algorithms for an efficient calculation of the dot product of SNP vectors. Motivated by the remarkable speed improvements of these implementations, a number of tailored algorithms for the dot product have been developed for different instruction set architectures which are up to 48 times faster than a naïve BLAS-based implementation (Schlather, 2023).

Additionally, some software solutions have studied the benefit of offloading genotype matrix operations to the GPU. PLINK 2.0 provides a BLAS-based calculation of the GRM on GPUs. However, according to the documentation, this functionality is just provided as a proof-of-concept. The Julia package SnpArrays.jl (Zhou et al., 2020) offers a pure-Julia solution for accelerating the multiplication of SNP matrices by a floating-point vector on GPUs.

Over the past few years, there has been a rising interest in low-precision arithmetics in the field of deep learning (Hubara et al., 2017), which has led to hardware improvements. For example, recent NVIDIA® architectures have introduced a number of new assembler instructions for this purpose. In deep learning, the method of quantization reduces the cardinality of possible values of a parameter by using low-precision integers and has been used in neural networks to increase the number of parameters (Gholami et al., 2022; Dettmers et al., 2022; Kim et al., 2022). This progress has opened new paths to explore for acceleration in genomic calculations.

We present the library miraculix which implements functions for the GPU-based multiplication of compressed SNP matrices by itself or floating-point matrices, which helps to accelerate the calculation of the GRM, LD and other essential quantities in genomics. In contrast to some of the aforementioned software packages such as PLINK, the package miraculix offers only a narrow, highly fine-tuned functionality and is designed to allow a neat integration into genomic analysis pipelines. Furthermore, it differentiates itself from other GPU software solutions by leveraging low-precision instructions available on NVIDIA® GPUs to operate on compressed SNP data. This technique reduces device memory requirements and is substantially faster than solutions in floating-point format. We provide interfaces which can be used by existing libraries for genomic analysis or in higher-level programming languages such as Julia (Bezanson et al., 2017).

Materials and Methods

For a diploid species, the SNP genotype matrix **Z** describes the genomic information of a set of genetic markers in the population. That is, $\mathbf{Z} \in \{0,1,2\}^{n \times k}$, where n is the number of individuals in the population and k is the number of SNPs. Due to the dramatic decrease in sequencing costs over the last decades, it is now possible to genotype millions of SNPs in vast populations or, alternatively, impute incompletely genotyped individuals. Therefore, researchers regularly deal with extraordinarily large data sets. For instance, the UK Biobank comprises broad genetic data of hundreds of thousands of human individuals (Bycroft et al., 2018). The SNP genotype matrix is used for a wide range of genomic analyses. For instance, the SNP genotype matrix is used for computing the VanRaden 1 GRM **G**, which is defined by

$$\mathbf{G} = \frac{\mathbf{PZ}(\mathbf{PZ})'}{2\mathbf{p}' \cdot (1_\mathrm{k} - \mathbf{p})}$$

with $1_k = (1, \ldots, 1)'$ denoting a vector of length k consisting only of 1s, $\mathbf{p}$ denoting the vector of allele frequencies and the matrix $\mathbf{P} = \mathbf{I} - 2 \cdot 1_n\mathbf{p}'$ for the identity matrix **I**. Here, the matrix **P** scales **Z** to have zero-centered allele counts (VanRaden, 2008). In genome-wide analysis studies (GWAS), the SNP genotype matrix $\mathbf{Z}$ is used to calculate regression coefficients of traits on one or multiple SNPs (Jiang et al., 2019). In the analysis of LD, the SNP genotype matrix is used to approximate the correlation statistic r^2 through the computation of the correlation between allele counts in **Z**. Due to the intrinsic properties of a SNP matrix, the efficient computation of $\mathbf{ZZ}'$ or $\mathbf{Z}'\mathbf{Z}$ is in fact the problem of a $\{0,1,2\}$-matrix multiplication (Chang et al., 2015; Schlather, 2023). Memory-efficient storage formats for **Z**, such as the PLINK 1 binary format, only use 2 bits per entry and the conceptual arrangement of these bits yields different multiplication approaches. A number of highly efficient SIMD-based algorithms for Central Processing Units (CPUs) have been suggested (Schlather, 2023; Chang et al., 2015). Here, we rely on an allele-count encoding for our GPU implementation MMAGPU, which stores counts in unsigned 2-bit integer format. This allows us to target the 4-bit matrix multiplication assembler instructions on modern NVIDIA® GPUs of compute capability 7.5 and higher. Through bit-masking and shift operations, we obtain a straightforward matrix multiplication microkernel. For fast data movement from global memory to shared memory to the cores and back, our library extends the CUTLASS library (NVIDIA, 2023) with 2-bit specializations, utilizing the available fast tile iterators. Since the resulting multiplication function is mainly bound by data transfers between the GPU and main memory, we divide the multiplication into blocks of rows and parallelize the multiplication of these rows into different threads and streams respectively.

Deviating from the above computations **ZZ'** and $\mathbf{Z}'\mathbf{Z}$, an efficient multiplication of the SNP genotype matrix by a floating-point matrix is required for other essential operations in genomics, e.g., in GWAS. Recently, we have presented functionality in miraculix for offloading this type of computation to GPUs and how this functionality can be used for accelerating single-step evaluations (Freudenberg et al., 2023).

To our knowledge, miraculix is the first software library which offers a GPU-based implementation of optimized matrix multiplications on compressed genotype data. The R packages MoBPS (Pook et al., 2020) and EpiGP (Vojgani et al., 2023), as well as the proprietary software MiXBLUP (ten Napel et al., 2021), have integrated miraculix.

Results & Discussion

Since multiplications of the SNP genotype matrix are an essential operation in a number of computational tasks in genomics, miraculix can be used as the backend for various calculations. In this section, we describe four possible applications of our high-performance GPU implementation and demonstrate how it enables the processing and analysis of datasets in previously unattainable computing times.

Genomic Relationship Matrix

For large dimensions of **Z**, a straightforward calculation of **G** becomes computationally prohibitive and a careful treatment of the involved operations is required. The decomposition

$$c\mathbf{G} = \mathbf{M} - 1_n\mathbf{p}'\mathbf{M} - \mathbf{M}1_\mathrm{n}\mathbf{p} + \mathbf{M}1_\mathrm{n}\mathbf{p}'\mathbf{p}1_\mathrm{n}\mathbf{M}$$

with $\mathbf{M} = \mathbf{ZZ}'$ and $c = 2\mathbf{p}'(1_\mathrm{k} - \mathbf{p})$, reveals that the matrix **G** can be obtained from **M** at relatively low computational costs of order $n^2 + nk$, whereas **M** requires $O(kn^2)$ (Schlather, 2023). In Figure 1, we compare the computation time of our GPU implementation with the CPU-targeted solution in PLINK. As these evaluations are performed on different hardware, we also benchmark a naive GPU implementation, which involves unpacking compressed genotype data into unsigned integers and uses the NVIDIA® cuBLAS library for multiplication. This approach resembles the proof-of-concept GPU solution implemented in PLINK, though we opted to store the input data in integers of 8 bits to save memory, while PLINK uses single-precision floating-point values. We simulated three different sets of genotype markers for a population of 22 000 individuals with the simulation utility in PLINK: A low-density array with 50 241 markers ("Low"), a medium-density array with 250 000 markers ("Medium") and a high-density one with 1 000 000 markers ("High"). For reference, the UK Biobank currently comprises about 850 000 directly measured variants. We tested the GPU functions on an NVIDIA GPU A100 with 80GB of device memory, while running PLINK on a dual-socket AMD® EPYC 7513 (2.6 GHz) with 32 dedicated cores each using the PLINK options *--make-rel square cov*. The results displayed are the median of 5 evaluations. Though direct conclusions on the efficiency of each solution are hard to draw due to the different hardware involved in the benchmarks, it can be observed that wall clock times in miraculix are smaller by a factor of at least 18 across the three test sets compared to PLINK. On the large dataset, the computation time was reduced from approximately 20 minutes to 56 seconds. Juxtaposing our solution to the simple cuBLAS-based solution, we see that significant speed gains can still be achieved by using our stack of microkernels for sub-byte integers and efficient memory management.

Yet, the significantly higher price tag of the A100 GPU has to be considered when evaluating these results: It is available at about 15 000 USD with a thermal power design (TDP) of 300W, while each of the two AMD® EPYC 7513 (2.6 GHz) CPUs has a recommended price of 2 840 USD with a TDP of 200W. Considering the power consumption of the evaluated methods, it is reasonable to assume that the GPU approaches are significantly more efficient than PLINK. Making a rough estimate based on the involved TDPs and computing times, a reduction in the magnitude of 20 in terms of power consumption can be presumed.

The gBLUP model

The genomic BLUP (gBLUP) model is widely used in population analysis to capture additive genetic effects (Misztal and Legarra, 2017) and is the basis for various extensions such as the extended gBLUP model, the single-step gBLUP model or the epistatic random regression BLUP model (Misztal et al., 2009; Jiang and Reif, 2015; Vojgani et al., 2021). In the gBLUP model, a quantitative trait $\mathbf{y}$ is assumed to be in a linear relationship with the genetic markers and environmental influences, captured in a matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$. The effects of SNPs are traditionally assumed to be random, resulting in the model

$$\boldsymbol{y} = \boldsymbol{Xb} + \boldsymbol{PZu} + \boldsymbol{e},$$

where $\mathbf{b}$ is a vector of fixed effects, $\mathbf{u} \sim N(0, \sigma_u^2\mathbf{I})$ is a vector of random effects and $\mathbf{e} \sim N(0, \sigma_e^2\mathbf{I})$ is an error term independent of $\mathbf{u}$. Alternatively, the term $\mathbf{g} = \mathbf{PZu}$ can be used to denote the breeding values. Then, $\mathbf{g}$ is normally distributed with mean 0 and covariance matrix $\sigma_g^2\mathbf{G}$ for $\sigma_g^2 > 0$. Furthermore, denoting $\mathbf{V} = \frac{\sigma_e^2}{\sigma_g^2}\mathbf{I} + \mathbf{G}$, the best linear unbiased estimator (BLUE) for $\mathbf{b}$ is given by

$$\widehat{\boldsymbol{b}} = \left(\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X}\right)^{-1}\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{y},$$

and the best linear unbiased predictor (BLUP) for $\mathbf{g}$ is given by

$$\widehat{\boldsymbol{g}} = \boldsymbol{GV}^{-1}\left(\boldsymbol{y} - \boldsymbol{X}\,\widehat{\boldsymbol{b}}\right).$$

In practice, the variance components σ_g^2 and σ_e^2 are either derived from previous estimates on the heritability of the trait $\mathbf{y}$ (e.g., by comparing offspring phenotypes with parental phenotypes) or estimated through Restricted Maximum Likelihood (REML), for instance, using the software package ASReml (Butler et al., 2017). Considering the above identities, the quantities $\hat{\mathbf{g}}$ and $\hat{\mathbf{b}}$ can be derived from the GRM $\mathbf{G}$ and estimates for σ_g^2 and σ_e^2 through a Cholesky decomposition. To this end, we utilized the cuSOLVER library to offload this computation to the GPU. In a recent empirical study, the full gBLUP calculation with miraculix showed an acceleration of up to 100 times compared to traditional software in the case where the heritability is known (Pook et al., 2021).

Additionally, two recent studies investigating the effects of epistasis utilized the efficiency of optimized CPU functions in miraculix (Vojgani et al., 2021, 2023). However, it should be noted that the memory requirements for setting up the GRM increase quadratically with the number of individuals which puts a limit to potential problem sizes.

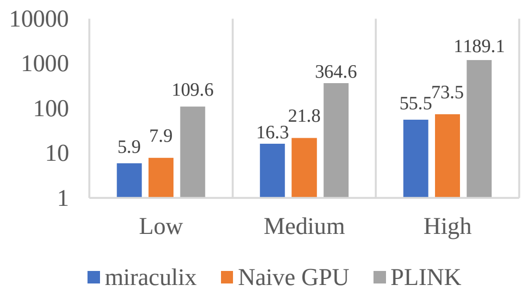

Figure 1. Wall clock times for the calculation of the GRM on three simulated sets of SNP genotypes.

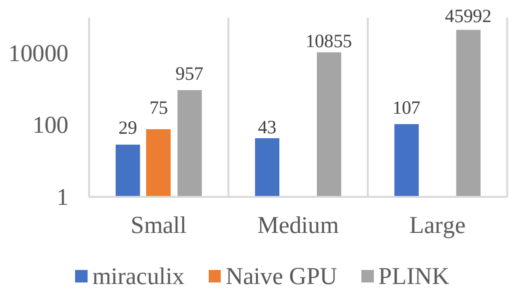

Figure 2. Wall clock times for the calculation of the LD on three simulated sets of SNP genotypes.

Linkage Disequilibrium

LD is a way of describing the dependence structure between pairs of alleles in a set of markers and there exist different statistics to capture this information in a population (Pritchard and Przeworski, 2001). The software PLINK implements the LD statistics r^2, D and D', which can be thought of as correlation measures between alleles. Though the true linkage value is based on haplotypes, it is sometimes approximated by the allele count correlations (e.g., in PLINK). That is, the squared correlation between the columns *i* and *j* of $\mathbf{Z}$ is used as value for r^2. Since the correlation matrix $\mathbf{R}$ can be written as

$$\boldsymbol{R} = \boldsymbol{D}^{-1/2}\,\widetilde{\boldsymbol{M}}\boldsymbol{D}^{-1/2}$$

for $\widetilde{\mathbf{M}} = \mathbf{Z}'\mathbf{Z} - 4n \cdot \boldsymbol{pp}'$ and $\mathbf{D} = \operatorname{diag}(\widetilde{\mathbf{M}})$, the matrix of pairwise r^2 values can be computed from $\widetilde{\mathbf{M}}$ at low cost. While the computation of $\mathbf{R}$ for a small block of SNPs with a limited number of individuals is straight-forward, a simple algorithm for the detection of LD between distant SNPs (so-called long-range LD) or the calculation of the average LD decay in a large part of the chromosome becomes cumbersome.

In our experiments, we calculated the matrix $\mathbf{R}$ of 50 241 markers across three simulated populations: A small population of 102 000 individuals (“Small”), a medium-sized one comprising 751 000 individuals (“Medium”) and a large population of 3 101 000 individuals (“Large”). As inflating the large population to single-precision floating-point values would require approximately 580GB of memory, this approach is impractical for LD calculation. Since miraculix processes SNP data in compressed format and subdivides the computation of the SNP matrix multiplication into blocks, only about 6 GB of device memory was required. Using the same hardware set-up as above, we compare our solution with the implementation in PLINK on 64 cores and a simple GPU solution in cuBLAS. However, due to its inherently higher device memory requirements, the latter could only be evaluated on the small dataset. Results are displayed in Figure 2 and are the median of 5 evaluations for the GPU functions. PLINK calculations were only performed once as wall clock times on these test sets made further evaluations unreasonable. For LD calculation, the PLINK options --*r square* were used. We observe that compute times in PLINK were more than 400 times higher on the large dataset.

Principal component analysis

For a column-wise standardized matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ the first m principal components (PCs) are defined by $\mathbf{X}\mathbf{v}_1, \ldots, \mathbf{X}\mathbf{v}_m$, where $\mathbf{v}_1, \ldots, \mathbf{v}_m$ solve the maximization problems

$$\max_{\mathbf{v}_1 \in \mathbb{R}^p, \|\mathbf{v}_1\|=1} \|\mathbf{X}\mathbf{v}_1\|$$

and

$$\max_{\mathbf{v}_i \in \mathbb{R}^p, \|\mathbf{v}_i\|=1, \mathbf{v}_1'\mathbf{v}_i=0, \ldots, \mathbf{v}_{i-1}'\mathbf{v}_i=0} \|\mathbf{X}\mathbf{v}_i\|$$

for $i = 2, \ldots, m$. Even though the transfer of the principal component analysis (PCA) to non-continuous data is not straightforward and a topic of ongoing research (see, e.g., Schlather and Reinbott (2021) for an approach to non-Euclidean data), PCA is still regularly used as an auxiliary tool in statistical genomics. Since PCA is a dimension-reducing method aimed at capturing large parts of variation in the dataset, PCs of the GRM are used in empirical studies in genetics to investigate population structure (e.g., by Steyn et al. (2022)) or as an auxiliary tool in the REML-based estimation of variance components (Thompson and Shaw, 1990; Lee and van der Werf, 2016). Principal components of the SNPs are regularly used to control for population stratification in GWAS studies or in breeding value estimation to reduce computational costs (Price et al., 2006). Popular software solutions include Eigensoft and PLINK (Price et al., 2006; Chang et al., 2015). Since PCA requires the multiplication of an orthonormal matrix of eigenvectors of $\mathbf{X}'\mathbf{X}$ by $\mathbf{X}$, miraculix can help to accelerate the PCs of a population by a fast computation of the GRM. Similarly, the PCs of SNPs can be derived from the $\mathbf{Z}'\mathbf{Z}$ matrix. However, if the dataset contains a lot of markers, constructing this matrix is challenging. The functionality of miraculix to multiply a SNP matrix by a floating-point matrix helps to alleviate this burden since there exists randomized algorithms for the singular value decomposition that do not require an explicit construction and calculate the first m eigenvalues and their corresponding eigenvectors with high accuracy (Halko et al., 2011). We provide an exemplary implementation in Julia.

gBLUP computing times

To evaluate the performance of miraculix in a practical setting, we simulated a population of $50\,000$ animals with $727\,605$ SNP variants based on the Illumina BovineHD BeadChip (Cunningham et al., 2021). Our supplementary Julia functions are linked to the interface of the library and perform low-cost post-processing operations on its return values. Emulating typical computational tasks in practice, we first load and process our data, which is stored in PLINK binary format on the disk, then calculate the GRM of the population and the SNP-wide PCs for later usage in inferring the parameters of the gBLUP model. PCs were modeled as fixed effects. Data processing operations were performed on an AMD® EPYC 7513 (2.6 GHz), while SNP matrix operations were offloaded to an NVIDIA® GPU A100. Since the CPU was mainly used for data preprocessing, we only used 8 dedicated cores. Due to the memory efficiency of our implementation, we were able to use the version of the A100 GPU with only 40GB of device memory. Computation of the GRM involved calculating the SNP matrix cross-product of dimensions $50\,000$ times $50\,000$ while retrieving the first 10 principal components required the multiplication of the SNP matrix by a floating-point matrix to obtain the approximate eigenvectors of SNP-wide covariance matrix. To estimate the vectors $\hat{\mathbf{b}}$ and $\hat{\mathbf{g}}$ of the gBLUP model, the Cholesky decomposition of the stretched GRM needed to be computed to solve the involved equation systems. Since heritability was assumed to be known, the ratio of variance components did not need to be estimated on the data.

Results are displayed in Table 1. We note that data processing now constitutes a significant portion of the total compute resource requirements both in terms of memory and computing times, as it requires a 2-bit format conversion and reordering of the bit-level values. The construction of the GRM was performed in just approx. 30 seconds, whereas the PCA calculation and the Cholesky decomposition needed 18 seconds and 13 seconds respectively. In total, approx. 36 gigabytes of main memory and 19 gigabytes of device memory were used.

Table 1. Computing times for various steps in a gBLUP calculation.

Calculation	Wall clock time (s)	Main memory usage (GB)	Device memory usage (GB)
Data set-up	20.20	26.24	-
GRM	30.48	18.67	5.57
PCA	17.63	5.01	17.23
Cholesky	12.73	0.13	18.67
Total	95	36.08	18.67

Computations were performed on a single NVIDIA® GPU A100-40GB using the Julia interface of miraculix. Total wall clock time includes additional system start-up time.

Conclusions

We have presented the capability of miraculix to offload essential operations on genomic data to the GPU. We illustrated its benefits in four applications. The approach outperforms existing CPU-based software solutions significantly and thereby enables much faster processing of genomic datasets of substantial size. Furthermore, it works on compressed data and therefore allows the processing of huge datasets. Overall, our experiments showed that a full gBLUP on a population of 50 000 individuals could be performed in little more than 1.5 minutes. Considering that a similar task was assumed to be computationally infeasible by VanRaden (2008) at the time, we find the performance improvements to be promising and encourage the use of GPUs to accelerate the processing of large datasets in genomics. While there is a suite of established methods to deal with extraordinary dimensions, e.g., Algorithm for Proven and Young (APY) (Misztal et al., 2014) or the use of iterative solvers (Strand´en and Lidauer, 1999), these approaches can similarly benefit from the techniques introduced in this article. To allow miraculix to handle further increases in dataset sizes, future software versions might include distributed calculations for high-performance clusters via a Message Passing Interface (MPI), which would extend its applicability to datasets that still cannot be fully stored in device memory. Furthermore, since the variance component estimation through REML is another computational bottleneck in genomic analyses, it would be interesting to offload this procedure to the GPU as well. Extensions of miraculix to AMD® or Intel® GPUs would be useful to allow researchers to take full advantage of existing computer hardware.

Acknowledgements

Analysis for this work was performed on the HoreKa and Helix compute clusters funded by the Ministry of Science, Research and the Arts Baden-Württemberg through bwHPC, by the German Federal Ministry of Education and Research and by the German Research Foundation (DFG). A.F. and M.S. acknowledge financial support by the Research Training Group ”Statistical Modeling of Complex Systems” funded by the German Science Foundation. The authors thank Ross Evans and the Irish Cattle Breeding Federation for providing the genomic datasets used in this study.

References

Amadeu, R.R., Cellon, C., Olmestead, J.W., Franco Garcia, A.A. and Resende Jr, M.F. (2016). AGHmatrix: R package to construct relationship matrices for autotetraploid and diploid species: a blueberry example. *Plant Genome*, 9(3), 1–10. URL https://pubmed.ncbi.nlm.nih.gov/27902800/

Barrett, J.C., Fry, B., Maller, J. and Daly, M.J. (2004). Haploview: analysis and visualization of LD and haplotype maps. *Bioinformatics*, 21(2), 263–265. URL https://doi.org/10.1093/bioinformatics/bth457

Bezanson, J., Edelman, A., Karpinski, S. and Shah, V.B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65–98. URL https://epubs.siam.org/doi/10.1137/141000671

Butler, D., Cullis, B., Gilmour, A., Gogel, B. and Thompson, R. (2017). ASReml-R reference manual version 4. VSN International Ltd, Hemel Hempstead, HP1 1ES, UK. URL https://asreml.kb.vsni.co.uk/wp-content/uploads/sites/3/ASReml-R-Reference-Manual-4.2.pdf

Bycroft, C. et al (2018). The UK Biobank resource with deep phenotyping and genomic data. *Nature*, 562(7726), 203–209. URL https://www.nature.com/articles/s41586-018-0579-z

Canela-Xandri, O., Rawlik, K., Woolliams, J.A. and Tenesa, A. (2016). Improved genetic profiling of anthropometric traits using a big data approach. *PLoS one*, 11(12), e0166755. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0166755

Chang, C.C., Chow, C.C., Tellier, L.C., Vattikuti, S., Purcell, S.M. and Lee, J.J. (2015). Second-generation PLINK: Rising to the challenge of larger and richer datasets. *GigaScience*, 4(1). s13742-015-0047-8. URL https://academic.oup.com/gigascience/article/4/1/s13742-015-0047-8/2707533

Cunningham, F. et al (2021). Ensembl 2022. *Nucleic Acids Res.*, 50(D1), D988–D995. URL https://academic.oup.com/nar/article/50/D1/D988/6430486

Dettmers, T., Lewis, M., Belkada, Y. and Zettlemoyer, L. (2022). GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 30318–30332. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c3ba4962c05c49636d4c6206a97e9c8a-Paper-Conference.pdf

Endelman, J.B. (2011). Ridge regression and other kernels for genomic selection with R package rrBLUP. *Plant Genome*, 4(3), 250–255. URL https://acsess.onlinelibrary.wiley.com/doi/10.3835/plantgenome2011.08.0024

Freudenberg, A., Vandenplas, J., Schlather, M., Pook, T., Evans, R. and Ten Napel, J. (2023). Accelerated matrix-vector multiplications for matrices involving genotype covariates with applications in genomic prediction. *Front. Genet.*, 14, 1220408. URL https://www.frontiersin.org/articles/10.3389/fgene.2023.1220408/full

Gazal, S. et al (2017). Linkage disequilibrium–dependent architecture of human complex traits shows action of negative selection. *Nat. Genet.*, 49(10), 1421–1427. URL https://www.nature.com/articles/ng.3954

Gholami, A., Kim, S., Zhen, D., Yao, Z., Mahoney, M. and Keutzer, K. (2022). A Survey of Quantization Methods for Efficient Neural Network Inference, in: Thiruvathukal, G.K., Lu, Y., Kim, J., Chen, Y., Chen, B. (Eds.), Low-Power Computer Vision, first ed. Chapman and Hall/CRC, New York, pp. 291–326.

Granato, I.S.C., Galli, G., Couto, E.G.D.O., Souza, M.B.E., Mendonça, L.F. and Fritsche-Neto, R. (2018). snpReady: A tool to assist breeders in genomic analysis. Mol. *Breed.*, 38, 1–7. URL https://link.springer.com/article/10.1007/s11032-018-0844-8

Halko, N., Martinsson, P.G. and Tropp, J. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53, 217–288. URL https://epubs.siam.org/doi/10.1137/090771806

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. and Bengio, Y. (2017). Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18(1), 6869–6898. URL https://www.jmlr.org/papers/volume18/16-456/16-456.pdf

Jiang, L., Zheng, Z., Qi, T., Kemper, K.E., Wray, N.R., Visscher, P.M. and Yang, J. (2019). A resource-efficient tool for mixed model association analysis of large-scale data. *Nat. Genet.*, 51(12), 1749–1755. URL https://www.nature.com/articles/s41588-019-0530-8

Jiang, Y. and Reif, J.C. (2015). Modeling epistasis in genomic selection. *Genetics*, 201(2), 759–768. URL https://academic.oup.com/genetics/article/201/2/759/5930119

Kim, H., Grueneberg, A., Vazquez, A.I., Hsu, S. and de los Campos, G. (2017). Will big data close the missing heritability gap? *Genetics*, 207(3), 1135–1145. URL https://academic.oup.com/genetics/article/207/3/1135/5930744

Kim, Y.J., Henry, R., Fahim, R. and Hassan, H. (2022). Who says elephants can't run: Bringing large scale MoE models into cloud scale production. In Fan, A., Gurevych, I., Hou, Y., Kozareva, Z., Luccioni, S., Moosavi, N.S., Ravi, S., Kim, G., Schwartz, R., and Rücklé, A. (Eds.), Proceedings of The Third Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), Abu Dhabi, United Arab Emirates (Hybrid), pp. 36–43. URL https://aclanthology.org/2022.sustainlp-1.6

Lee, S.H. and van der Werf, J.H.J. (2016). MTG2: an efficient algorithm for multivariate linear mixed model analysis based on genomic information. *Bioinformatics*, 32(9), 1420–1422. URL https://academic.oup.com/bioinformatics/article/32/9/1420/1744518

Misztal, I., Legarra, A. and Aguilar, I. (2009). Computing procedures for genetic evaluation including phenotypic, full pedigree, and genomic information. *J. Dairy Sci.*, 92(9), 4648–4655. URL https://www.sciencedirect.com/science/article/pii/S0022030209707921

Misztal, I. and Legarra, A. (2017). Invited review: Efficient computation strategies in genomic selection. *Animal*, 11(5), 731–736. URL https://www.sciencedirect.com/science/article/pii/S1751731116002366

Misztal, I., Legarra, A. and Aguilar, I. (2014). Using recursion to compute the inverse of the genomic relationship matrix. *J. Dairy Sci.*, 97(6), 3943–3952. URL https://www.sciencedirect.com/science/article/pii/S0022030214002240

Mrode, R.A. (2014). Linear models for the prediction of animal breeding values, second ed., Cabi.

NVIDIA (2023). Parallel Thread Execution ISA – Application Guide. NVIDIA Corporation and Affiliates, v8.2 edition. Last accessed 17th of July 2023. URL https://docs.nvidia.com/cuda/parallel-thread-execution/index.html

Pallares, L. (2019). Searching for solutions to the missing heritability problem. *eLife*, 8, e53018. URL https://elifesciences.org/articles/53018

Pook, T., Schlather, M. and Simianer, H. (2020). MoBPS - Modular Breeding Program Simulator. *G3 (Bethesda)*, 10(6), 1915–1918. URL https://www.g3journal.org/content/10/6/1915

Pook, T., Reimer, C., Freudenberg, A., Büttgen, L., Geibel, J., Ganesan, A., Ha, N.T., Schlather, M., Mikkelsen, L.F. and Simianer, H. (2021). The Modular Breeding Program Simulator (MoBPS) allows efficient simulation of complex breeding programs. *An. Prod. Sci.*, 61, 1982–1989. URL https://www.publish.csiro.au/an/pdf/AN21076

Price, A., Patterson, N., Plenge, R., Weinblatt, M., Shadick, N. and Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. Nat. Gen., 38(8), 904–909. URL https://www.nature.com/articles/ng1847

Pritchard, J.K. and Przeworski, M. (2001). Linkage disequilibrium in humans: models and data. *Am. J. Hum. Genet*., 69(1), 1–14. URL https://www.sciencedirect.com/science/article/pii/S0002929707614396

Rohde, P.D., Fourie Sørensen, I. and Sørensen, P. (2019). qgg: an R package for large-scale quantitative genetic analyses. *Bioinformatics*, 36(8), 2614–2615. URL https://academic.oup.com/bioinformatics/article/36/8/2614/5688744

Schaeffer, L. (2006). Strategy for applying genome-wide selection in dairy cattle. *J. Anim. Breed. Genet*., 123(4), 218–223. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1439-0388.2006.00595.x

Schlather, M. and Reinbott, F. (2021). A semi-group approach to principal component analysis. *arXiv*. URL https://arxiv.org/abs/2112.04026

Schlather, M., Freudenberg, A., Moerkotte, G., Pook, T. and Vandenplas, J. (2023). Software project 'miraculix': Efficient computations with large genomic datasets. *Interbull Bulletin*, 59,15-22.

Singh, R.K. and Prasad, M. (2021). Big genomic data analysis leads to more accurate trait prediction in hybrid breeding for yield enhancement in crop plants. *Plant Cell Rep*., 40(10), 2009–2011. URL https://link.springer.com/article/10.1007/s00299-021-02761-x

Steyn, Y., Masuda, Y., Tsuruta, S., Lourenco, D., Misztal, I. and Lawlor, T. (2022). Identifying influential sires and distinct clusters of selection candidates based on genomic relationships to reduce inbreeding in the US Holstein. *J. Dairy Sci*., 105(12), 9810–9821. URL https://www.sciencedirect.com/science/article/pii/S0022030222005896

Strandén, I. and Lidauer, M. (1999). Solving large mixed linear models using preconditioned conjugate gradient iteration. *J. Dairy Sci*., 82(12), 2779–2787. URL https://www.sciencedirect.com/science/article/pii/S0022030299755359

ten Napel, J., Vandenplas, J., Lidauer, M.H., Strandén, I., Taskinen, M., Mäntysaari, E.A., Calus, M.P. and Veerkamp, R.F. (2021). MiXBLUP 3.0.1 manual. Animal Breeding and Genomics, Wageningen University & Research, Wageningen, the Netherlands, v3.0 edition. URL https://www.mixblup.eu/download.html

Thompson, E.A. and Shaw, R.G. (1990). Pedigree analysis for quantitative traits: variance components without matrix inversion. *Biometrics*, 46(2), 399–413. URL http://www.jstor.org/stable/2531445

VanRaden, P. (2008). Efficient methods to compute genomic predictions. *J. Dairy Sci*., 91(11), 4414–4423. URL https://www.sciencedirect.com/science/article/pii/S0022030208709901

Vojgani, E., Pook, T., Martini, J., Hölker, A., Mayer, M., Schön, C.C. and Simianer, H. (2021). Accounting for epistasis improves genomic prediction of phenotypes with univariate and bivariate models across environments. *Theor. Appl. Genet*., 134, 2913–2930. URL https://link.springer.com/article/10.1007/s00122-021-03868-1

Vojgani, E., Hölker, A., Mayer, M., Schön, C.C., Simianer, H. and Pook, T. (2023). Genomic prediction using information across years with epistatic models and dimension reduction via haplotype blocks. *PLoS one*, 18, e0282288. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0282288

Wainschtein, P., et al (2022). Assessing the contribution of rare variants to complex trait heritability from whole-genome sequence data. *Nat. Genet*., 54(3), 263–273. URL https://www.nature.com/articles/s41588-021-00997-7

Yang, J., Lee, S.H., Goddard, M.E. and Visscher, P.M. (2011). GCTA: a tool for genome-wide complex trait analysis. *Am. J. Hum Genet*., 88(1), 76–82. URL https://www.sciencedirect.com/science/article/pii/S0002929710005987

Yao, Z. (2020). LDkit: a parallel computing toolkit for linkage disequilibrium analysis. *BMC Bioinformatics*, 21(1), 461. URL https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-03754-5

Zhao, Y. et al (2021). Unlocking big data doubled the accuracy in predicting the grain yield in hybrid wheat. *Sci. Adv*., 7(24), eabf9106. URL https://www.science.org/doi/full/10.1126/sciadv.abf9106

Zheng, X., Levine, D., Shen, J., Gogarten, S., Laurie, C. and Weir, B. (2012). A high-performance computing toolset for relatedness and principal component analysis of SNP data. *Bioinformatics*, 28(24), 3326–3328. URL https://academic.oup.com/bioinformatics/article/28/24/3326/245844

Zhou, H. et al (2020). OpenMendel: cooperative programming project for statistical genetics. *Hum. Genet*., 139(1), 61–71. URL https://link.springer.com/article/10.1007/s00439-019-02001-z