GWA-X: An Extensible GPU Accelerated Package for Permutation Testing in Genome-Wide Association Studies

Mulya Agung and Albert Tenesa
University of Edinburgh

Abstract

Genome-wide association studies (GWAS) aim to identify genetic variants that are associated with a trait or disease. The scale of genomic datasets has increased to millions of genetic variants and hundreds of thousands of individuals, opening the possibilities for GWAS discoveries. However, large-scale GWAS analyses are prone to high false positive rates because of the multiple testing problem. Permutation testing is the gold standard for maintaining false positive rates, yet it is impractical for large-scale GWAS because it demands vast computational resources. This paper presents GWA-X, a software package that can exploit GPU potential to accelerate permutation testing in GWAS. Unlike previous methods, GWA-X employs a novel whole-genome regression method and a permutation testing strategy to batch the computations of many genetic markers. It achieved a two-order magnitude speed-up compared with the existing CPU-based and GPU-based permutation methods. In addition, it provides an extensible package for GWAS permutation testing on GPUs.

1 Introduction

Genome-wide association studies (GWAS) scan a genome-wide set of genetic variants or markers in a group of individuals to identify variants associated with a trait or disease. The growth of GWAS over the past decade has identified numerous associations for complex human traits and diseases. In recent years, the scale of genomic datasets has increased to millions of genetic variants and hundreds of thousands of individuals, opening the possibility for discoveries from GWAS and omics studies [1, 2].

Despite the availability of large genomic data, large-scale GWAS demands vast computational resources due to the many variants that need to be tested [3, 4]. In addition, conducting GWAS with millions of genetic variants has a severe problem of the potentially high number of false positive results due to many variants tested [5, 6] and non-normal phenotypic distributions [7, 8].

Permutation tests are the gold standard for multiple testing correction in GWAS because they can reduce false positives while maintaining statistical power [9–11]. However, permutation tests are computationally demanding because they require many statistical tests to obtain a statistical distribution. The growth of scale in genomic datasets further increases the computational burden of permutation tests. The scale is expected to grow to billions of variants and millions of people in the future [1, 12]. Consequently, permutation testing is hardly practical in large-scale GWAS.

The end of Dennard scaling and the imminent end of Moore's law have led to the necessity of using specialized computing processors such as graphics processing units (GPUs) to improve computational efficiency [13–15]. Because GPUs can perform numerical calculations faster than central processing units (CPUs) at comparable energy costs, they could significantly reduce GWAS's computational time and energy consumption. Although GPUs have been widely used in AI research [16], the use of GPUs in GWAS is still limited because existing GWAS tools that work well for CPUs cannot take advantage of GPUs due to hardware GPU constraints that are not present in CPUs. In addition, not all computations in GWAS permutation testing are suitable for GPUs. A collaboration between CPU and GPU is necessary to achieve high computational efficiency [17].

Several linear algebra libraries have been developed for GPUs [18, 19]. However, these libraries cannot efficiently parallelize GWAS' regressions because they aim to compute a regression of a large matrix problem in parallel to use all the GPU cores [20]. The regression in GWAS is computed for each variant or a group of variants, resulting in many regression computations of small matrices instead of one large matrix. Thus, these libraries cannot efficiently utilize GPUs and cannot achieve significant performance improvements compared to CPUs.

Besides the computational demand, diverse permutation testing approaches have been developed for different applications in GWAS [10, 21, 22] simultaneously with novel regression methods to account for the population structure in

the emerging genomic data [3, 23]. Meanwhile, the growth of genomic data size has imposed new computing system requirements, such as biobank platforms [2, 24]. GWAS permutation testing could benefit from the latest methods and computing systems. However, existing GWAS and permutation testing tools cannot be easily extended due to their architecture limitations. Therefore, there is a demand for an extensible package for GWAS permutation testing that can adapt to new developments.

This paper presents GWA-X, a software package that leverages GPUs to accelerate GWAS permutation testing. In contrast to previous methods, GWA-X batches regression computations of many genetic variants to achieve substantial acceleration on GPUs. The **contributions** of this paper are as follows.

- A whole-genome regression method for GPUs while accounting for population structure, including a genetic marker-based batch computation algorithm and a chunked data processing technique to handle large genotype data and use multiple GPUs.
- A permutation testing strategy for GPUs, including a memoization technique and a CPU-GPU collaboration strategy to accelerate permutation tests.
- An extensible and scalable software package that allows the addition of new regression models, permutation testing approaches, and computing system requirements.

2 Related Work

Several libraries have been developed to accelerate batch matrix computations on GPUs [19, 20, 25], which mostly focus on square matrix problems or independent tall-skinny matrices. However, a GWAS analysis requires regressions of many matrices of genetic markers and the same matrices of the trait and covariates. These libraries cannot parallelize many regressions enough to maximize GPU occupancy. Hence, the computational performance of these libraries will not scale with many genetic markers in GWAS.

Most of the existing GWAS permutation testing methods focused on CPUs and cannot use GPUs [9, 26–28]. More recent methods use GPUs to accelerate the permutation tests in GWAS [10, 29]. However, these methods rely on existing GPU libraries that cannot fully utilize GPU's potential performance. In addition, the previous GPU-based methods cannot handle datasets larger than GPU memory and are not extensible to different permutation testing approaches and computing system requirements.

In contrast with the GPU libraries and the previous methods, GWA-X batches regression computations of many genetic markers and uses a chunked data processing technique to handle large datasets. The evaluation of this paper demonstrates the superiority of GWA-X over the previous CPU-based and GPU-based methods and an implementation using the current state-of-the-art GPU library.

3 Method

Typical GWAS approaches perform generalized linear models (GLMs) to test associations of genetic variants or markers and traits depending on whether a continuous or binary trait is analyzed [5]. For each genetic marker, a GLM regression is given by the following equation:

$$Y = \alpha C + \beta G + \varepsilon, \tag{1}$$

where Y is a trait or disease of interest, G is a genetic marker, C represents a matrix of covariates, and ε is the residual.

To account for sample relatedness or population structure, an additional step is required to reduce the bias caused by genetic relatedness among individuals, such as in linear mixed models (LMMs) [4, 11]. However, LMM requires the computation of the genetic relationship matrix (GRM), which is computationally expensive with large samples [30]. Alternatively, ridge regression is a machine learning approach that can estimate the sample relatedness and is more computationally efficient [3, 31]. In addition, it allows population structure correction to be decoupled from association testing and permutation tests.

GWA-X adopts the previous method [3] to compute the genetic predictors correlated to the trait, $\hat{y}^*_{pred}$, that estimates the sample relatedness. The predictors are obtained by fitting the model in Equation (2) using ridge regression across a range of shrinkage parameters, where W is a matrix derived from G with substantially fewer columns, and η is estimated as $\widehat{\eta}$ by adding an $L2$ penalty to the log-likelihood function of the model in Equation (2). After $\widehat{\eta}$ has been estimated, the predictors can be obtained as in Equation (3), using the leave-one-chromosome-out (LOCO) scheme by ignoring columns of the chromosome containing the genetic marker being tested when calculating the predictors. Then, instead of using the original trait values (Y) for Equation (1), GWA-X uses the corrected trait values ($\hat{Y}$) obtained from removing the predictors from Y as in Equation (4). More details on the ridge regression method can be found in the original paper [3].

$$Y = W\eta + \varepsilon. \tag{2}$$

$$\hat{y}^*_{pred} = W\widehat{\eta}. \tag{3}$$

$$\hat{Y} = Y - \hat{y}^*_{pred}. \tag{4}$$

In GWAS permutation testing, the trait values are permuted Q times while keeping the variants intact to account for linkage disequilibrium (LD), and the statistical tests are repeated

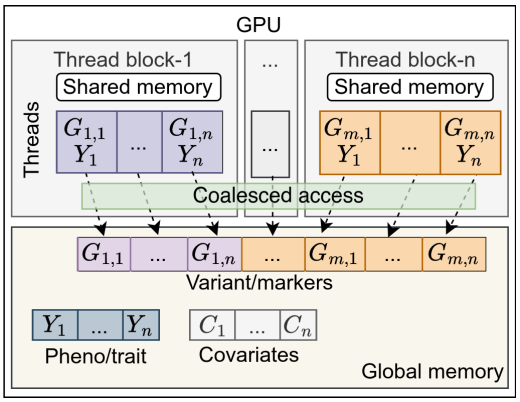


Figure 1: GWA-X's thread organization on GPU

for each permuted dataset [9, 22, 27, 29]. The test results are then used to generate the distribution under the null hypothesis of no association. Statistical significance (p-value) of the permutation testing (P_{perm}) can be calculated by Equation (5), which is the fraction of permuted datasets for which the value of the test statistic (T_q) is equal or greater to the value observed on the original data (T_{obs}).

$$P_{perm} = \frac{\sum_{q=1}^{Q} {}^{1}(|T_q| \geq |T_{obs}|)}{Q}. \quad (5)$$

Accordingly, the number of regression computations will increase linearly with the number of genetic markers and permutations. Regressions with many variants and large sample sizes are computationally demanding even without permutation [3, 4]. Therefore, accelerating the regressions is key to achieving a high computational efficiency in permutation tests.

GWA-X accelerates the computational performance of GWAS permutation testing in two steps: it increases the degree of parallelization for regression computations on GPUs. Then, it employs a CPU-GPU collaboration strategy to accelerate permutation tests. The following subsections detail these steps.

3.1 Massively Parallel Regression on GPUs

GPUs have two memory types: on-chip and global memories. On-chip memory is faster but much smaller than global memory. In GPU programming, computation tasks are executed as units, called threads, organized into a grid of thread blocks. Each thread block can share data via an on-chip memory space known as shared memory, which is typically small and can only be accessed by threads within the same block. Before the computation, threads load the data from the global memory to the on-chip memory. Unlike CPUs, adjacent threads must access adjacent data elements in global memory to achieve high speeds on GPUs. That is why methods designed for CPUs often perform poorly on GPUs [32].

GPUs have thousands of processing units that can carry out many calculations in parallel, which is vital to accelerating regression computations in GWAS. A regression method must maximize the number of processing units used in parallel to achieve high computational efficiency, whereas memory access is slower than computation. GWA-X increases the memory access throughput from/to GPU's global memory by coalesced access to maximize the use of GPU's memory bandwidth [20, 32].

Another key aspect is the organization of threads because it determines how many computation tasks can run in parallel and share data through shared memory. GWA-X uses the grid shown in Figure 1, which allows different markers and samples to be computed in parallel. Simultaneously, to use the fast shared memory, the threads of the same block compute the regression for the same marker. For example, regression stores temporary data on shared memory for matrix multiplication to reduce the number of accesses to global memory.

In GLM, the genetic and fixed effects (β, α) can be calculated using Equation (6), where X is a concatenated matrix consisting of a genotype vector with a covariate matrix ($G|C$) and Y or $\hat{Y}$ as the vector of trait values. Thus, the regression requires an matrix inverse operation, which is computationally intensive. GWA-X accelerates regressions by computing matrix inverse of many markers in parallel. It uses an algorithm extended from the previous parallel Gauss-Jordan algorithm [33]. The extended algorithm computes the matrix inverse of different genetic markers on different thread blocks, and the matrix inverse of a genetic marker is computed by all the threads of the same thread block to use shared memory.

$$\{\beta, \alpha\} = (X^T X)^{-1} X^T \hat{Y}. \quad (6)$$

GPUs have limited memory capacity, and the data required might not fit entirely into the GPU memory, such as in whole-genome regression. GWA-X addresses this problem by chunking genotype data for data transfers and processing, including decompressing compressed genotypes on GPUs to increase GPU occupancy. The chunk or block size can be specified by users according to the GPU memory size. As sequential data transfers to process all the chunks can delay GPU computations, GWA-X reduces the delay by overlapping the data transfers with GPU computations (Figure 2). In a system with multiple GPUs, the genotype chunks are distributed to the GPUs, and regressions are computed in parallel on multiple GPUs (Figure 3).

3.2 Permutation Testing Strategy

CPUs and GPUs have unique features and strengths. Hence, dividing computational workloads among them can significantly improve computational efficiency [17]. For instance, data shuffling involves much more memory operations than numerical calculations. Hence, permuting trait values of large samples can be slower on GPUs than on CPUs due to the

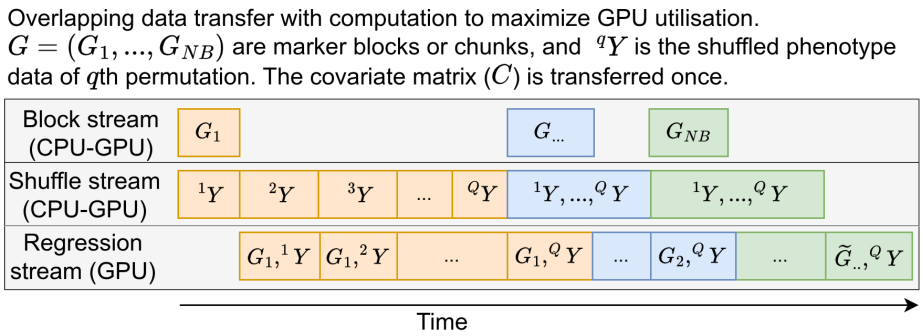


Figure 2: Overlapped computations with data transfers

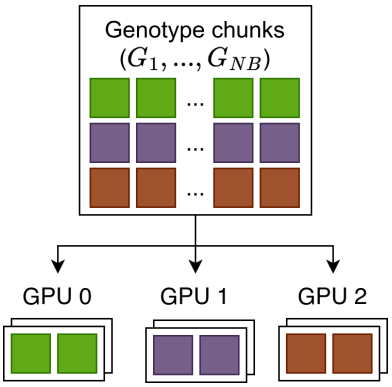


Figure 3: Genotype chunk distribution on multiple GPUs

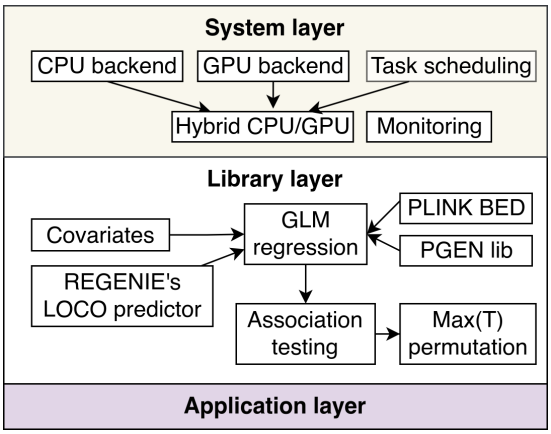


Figure 4: The software architecture of GWA-X

GPU's memory access latencies. In addition, transferring the permuted data from/to CPUs and GPUs sequentially can delay GPU computations. GWA-X shuffles the trait data on CPUs and transfers them with chunked genotype data to GPUs overlapped with regression computations, as shown in Figure 2.

In permutation tests, trait values (Y or $\hat{Y}$) are shuffled for each permutation q, denoted as (${}^q\hat{Y}$). Then, the β and α coefficients are calculated for each permutation (β_q, α_q) using Equation (7). The permutation test statistic (T_q) for a genetic marker is calculated using Equation (8), where β_q and SE_q are the coefficient and standard error of the permutation q, respectively. Therefore, the genotype covariance matrix, $(X^TX)^{-1}$, needs to be computed only once and can be reused for the whole permutation tests, including for the SE calculation.

$$\{\beta_q, \alpha_q\} = (X^TX)^{-1}X^T({}^q\hat{Y}). \tag{7}$$

$$T_q = \beta_q / SE_q. \tag{8}$$

Instead of computing standard errors (SEs) sequentially for each marker, GWA-X calculates them for a batch of markers at once on GPUs. For each marker, SEs are calculated using the following matrix algebra:

$$\begin{aligned} C &= (X^TX)^{-1} * MS_{res}, \\ SE &= \sqrt{c_{0,0}}, \end{aligned} \tag{9}$$

where C is the parameter covariance matrix, $c_{0,0}$ is the value from C's diagonal vector that corresponds to β, and MS_{res} is the mean square of residuals for each permuted dataset. GWA-X applies a memoization technique that stores and reuses the covariance matrices of genetic markers to reduce computation time.

4 GWA-X Architecture

GWA-X's architecture, shown in Figure 4, comprises (1) a system layer containing computing system backends, (2) a library layer implementing regression models and permutation testing approaches, (3) and an application layer to integrate the library and system layers.

As the size of genomic data will increase in the future [2], the use of distributed computing systems is necessary. Accordingly, the permutation methods must be scalable to distributed CPU-GPU systems, imposing new system requirements. The

system layer allows additional backends to use different systems, such as multiple GPUs and distributed systems, without changing the library layer. This flexibility also broadens GWA-X applicability to various computing systems, such as biobank platforms [2, 24]. Besides system backends, the system layer provides a task scheduler to optimize parallel executions on CPUs and GPUs and a monitoring subsystem to track computational performance and energy usage.

The library layer contains implementations of regression algorithms and permutation testing approaches with a programming interface to reuse them. It presents the advantage that the regression and permutation testing routines can also be used for different analyses to benefit from GPUs, such as multi-omics [34] and rare variant analysis [22]. The application layer integrates the library and system components and provides the user interface for conducting permutation tests.

4.1 Implementation

GWA-X is an active open-source project[†] developed at the University of Edinburgh, UK. It fully integrates with the Python environment and is easy to install using the Pip package manager. The regression method is built from scratch and compiled into machine code using the Numba compiler [35] to achieve comparable performance to a compiled programming language.

At the time of writing, a whole-genome regression method for quantitative traits and a standard permutation testing approach have been implemented. For accounting for population structure, GWA-X can use the LOCO prediction file generated from Step 1 of the REGENIE tool [3] and a matrix of covariates. It supports the common PLINK BED and PGEN genotype formats [28] and CSV formatted data of phenotype or trait.

As a future roadmap, GWA-X will support logistic regression for binary traits, adaptive permutation [21], and system backends to use distributed CPU-GPU systems. GWA-X currently requires REGENIE for computing ridge regression to generate the genetic predictors. Thus, another future work is extending GWA-X to support ridge regression on GPUs. The extensible architecture will allow future development to be applied modularly, requiring no or minimal modification to the implemented parts and different layers.

5 Data and Simulation

The statistical and computational performances of GWA-X are evaluated on simulated genotype-trait datasets. To mimic the linkage disequilibrium structure of real datasets, the genotype data were simulated from the HapMap3 (release 2 build 36) haplotype data using HAPGEN2 [36], resulting in up to 430,000 samples and 1,002,365 array SNPs with minor allele count higher than 5.

[†]https://git.ecdf.ed.ac.uk/magung/gwa-x

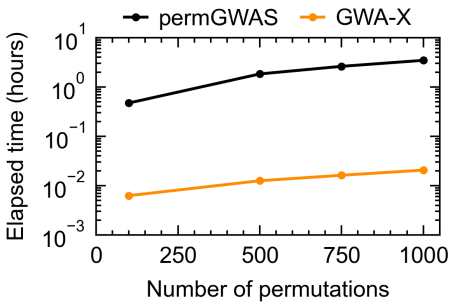


Figure 5: The computational performance of GWA-X and permGWAS with 4,000 samples and one million markers

Artificial traits were simulated for 5,000 and 10,000 causal SNPs as causal markers from the genotype simulated from HapMap3. The effect sizes for the causal SNPs (β) were sampled from a normal distribution with a mean of zero, where the variance was determined based on the desired proportion of trait variance explained by the causal SNPs, h^2. The effect from population structure (α) was set so that the proportion of the trait variance explained by the top principal component was 10%. Random noise (ε) was generated from a normal, Student's t, or gamma distribution. The genetic predictors for the simulated traits were generated using REGENIE.

6 Evaluation

The evaluation comprises computational efficiency and statistical performance comparisons on the simulated datasets.

6.1 Computational Efficiency

A relatively recent method, permGWAS [10], has been proposed for accelerating permutation testing on GPUs. In contrast to GWA-X, permGWAS uses the PyTorch library [19] to perform batch-wise matrix computations on GPUs. Figure 5 shows the computation times of both methods on an NVIDIA A5000 GPU card with a simulated dataset of 4,000 samples and one million genetic markers, a quantitative trait, and a matrix of covariates. GWA-X outperforms permGWAS and achieved more than two orders of magnitude speed-up in permutation tests (129.9x on average), reducing the computational time from 3.4 hours to 74 seconds. The evaluation could not run permGWAS on larger samples because it requires a large memory space to load whole genotype data into memory. This fact shows the necessity of the data chunking technique in GWA-X to allow permutation tests with data sizes larger than CPU and GPU memories.

To investigate computational efficiency on larger samples, the evaluation compares the computational performance and

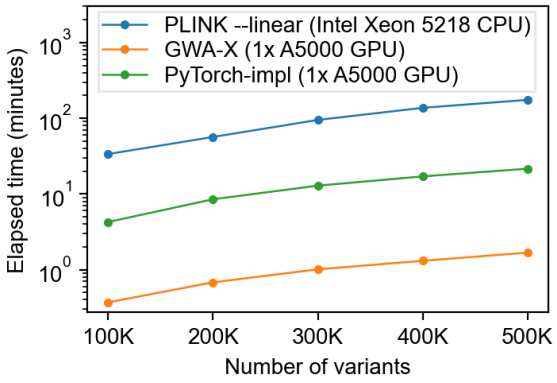


Figure 6: The computational performance of regression for quantitative trait analysis

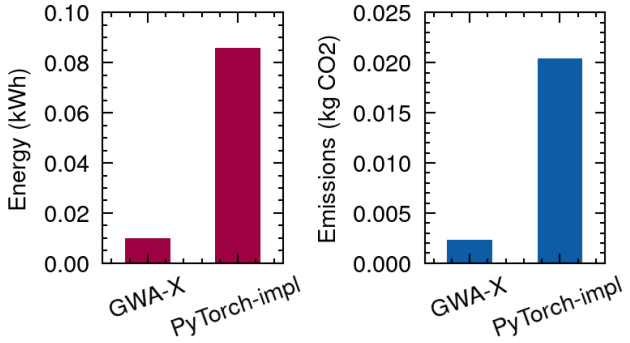


Figure 7: The estimated energy usage and carbon emission of GWAS regression

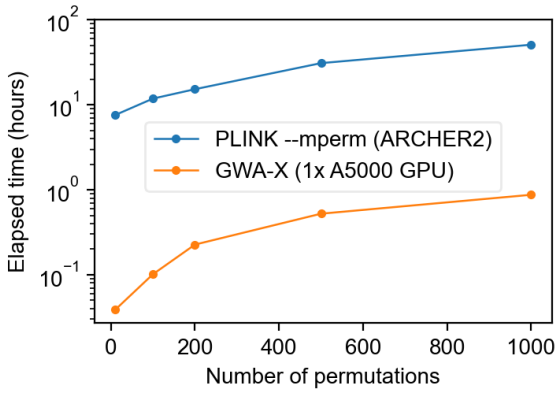


Figure 8: The computational performance of permutation tests with GWA-X and PLINK

energy consumption of GWA-X to a CPU-optimized software, PLINK [28], and an implementation using the GPU-accelerated PyTorch [19] library (PyTorch-impl), which uses the torch.linalg module to perform batch matrix computation on GPUs. In contrast with permGWAS, PyTorch-impl uses genotype data chunking to handle large samples. PyTorch-impl is available in the GWA-X project repository. The evaluation uses the simulated dataset of a quantitative trait, a covariate matrix, and a genotypes with 430,000 samples and up to 500,000 genetic markers.

GWA-X and PyTorch-impl were run on an NVIDIA A5000 GPU card, while PLINK was run with the `-linear` option on a 16-core Intel Xeon 5128 CPU @2.3 GHz. For permutation tests, GWA-X performance is compared with PLINK (`-linear mperm`) on a single node of the ARCHER2 supercomputer [37], equipped with 128-core AMD EPYC 7742 processors.

Figure 6 shows the computational performance of regression. GWA-X achieved a faster regression by a hundredfold compared with PLINK (95.5x on average) using only one GPU card, more than tenfold faster than the PyTorch-based implementation (12.5x on average). Figure 7 shows the energy consumptions and carbon emissions of GWA-X and PyTorch-impl, measured using CodeCarbon [38] that is implemented in GWA-X's system layer. GWA-X reduced energy and CO2 emissions by 88.4% from faster regressions.

For permutation tests, GWA-X achieved a speed-up ratio of more than two orders of magnitude (99.5x on average) compared with PLINK on ARCHER2, as shown in Figure 8. The results suggest that GWA-X can save substantial cost and energy by replacing hundreds of CPU nodes with a GPU card. This speed-up ratio is also much faster than the speed-up reported by the latest GPU-based methods [10, 29], demonstrating the effectiveness of GWA-X's regression method and permutation testing strategy.

To evaluate performance scalability on multiple GPUs, GWA-X was run on a system equipped with two NVIDIA A5000 GPU cards. Figure 9 shows GWA-X's runtimes with 500,000 genetic markers and 430,000 samples, and the number of permutations ranged from 1000 to 5000. For the same number of permutations, the runtime decreases almost linearly with the number of GPUs, and the overhead is only a small fraction (less than 0.82%) of the total runtime. The small overhead demonstrates the effectiveness of overlapping data transfers with computations. These results show that GWA-X's computational performance is scalable to multiple GPUs.

6.2 Statistical Performance

GWA-X's statistical performance is evaluated by comparing the empirical power and false discovery rate (FDR) of the standard genome-wide threshold ($5e-8$) and two permutation-based thresholds at the 5% level obtained from 1,000 and 10,000 permutations (Q). For the comparison, 10,000 causal markers ($h^2 = 0.2$) were simulated on the same dataset used in the computational performance evaluation with the random noise generated from a normal distribution. The significant

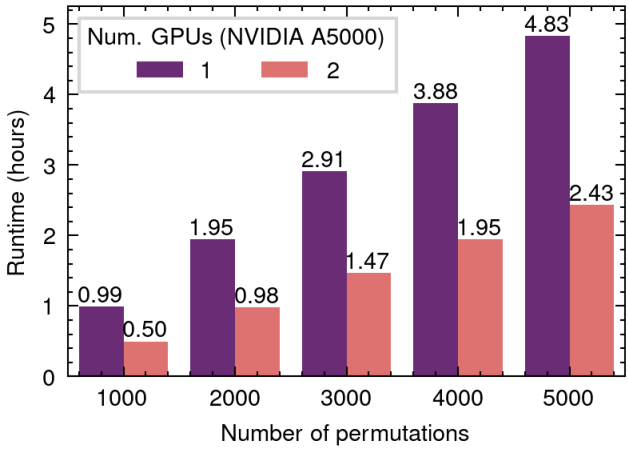


Figure 9: GWA-X's performance scalability on multiple GPUs

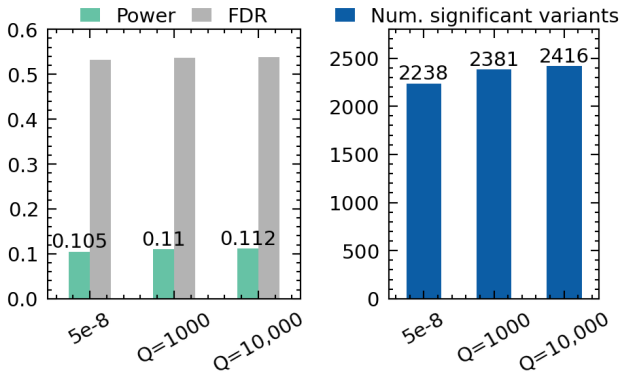


Figure 10: The empirical power, FDR, and number of significant markers of permutation-based thresholds

markers are the markers with p-values less than the threshold. The empirical power is the proportion of causal markers detected as significant, and FDR is the proportion of significant markers that are non-causal.

Figure 10 shows the power, FDR, and the number of significant variants detected using the standard and permutation-based significant thresholds. Compared with the standard threshold, the permutation-based thresholds detected more significant variants and achieved higher statistical power while maintaining FDR. The power increased with Q, indicating that the accuracy increased with the number of permutations. These results also show the benefit of GWA-X's acceleration, which allows more permutations to achieve higher accuracy.

Previous studies [7, 10] show that GWAS with non-normally distributed traits are prone to false positive associations. This evaluation compares the statistical performance of permutation-based and Bonferroni-adjusted thresholds on the simulated dataset of ~1M markers with 5,000 causal markers ($h^2 = 0.2$) and random noise generated from Student's t and gamma distributions. The Student's t-distributed noise (Std-t) was generated with a degree of freedom parameter of 4, and gamma-distributed noise was generated with shape parameters of 1 (gamma-1) and 2 (gamma-2). The permutation-based thresholds were the thresholds at the 5% level from 1,000 permutations.

Figure 10 shows the false positive rate (FPR) and power results of permutation-based and Bonferroni thresholds on the dataset with the non-normal noise distributions. FPR is the proportion of non-causal markers detected as significant. Permutation-based thresholds reduced FPR while maintaining power, with a higher FPR reduction shown in traits with gamma-distributed noise. For the Student's t-distributed noise, permutation-based thresholds showed marginal FPR reductions over Bonferroni thresholds, indicating that permutation-based thresholds become more stringent when the trait distribution becomes more skewed from normal, as also suggested by other recent work [10]. These results also show the benefit of permutation testing in reducing false positive associations in non-normally distributed traits.

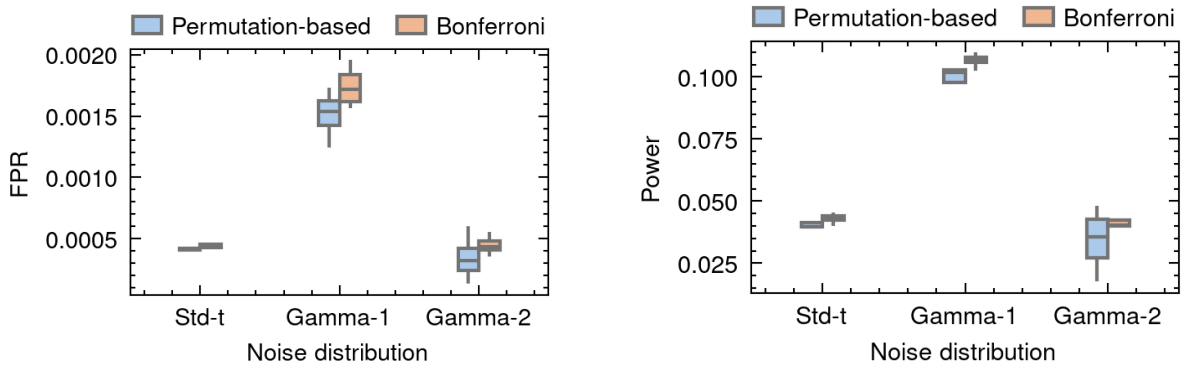
7 Discussion and Conclusion

Currently, there is no GPU-accelerated software for GWAS permutation testing that is both efficient and extensible. GWA-X employs a novel whole-genome regression method and a CPU-GPU collaboration strategy to fully utilize GPU computing capabilities. The computational efficiency and extensibility achieved simultaneously by its architecture are essential for emerging data scales and analyses in GWAS. The evaluation demonstrates unprecedented acceleration and energy saving on regression and permutation tests. Thus, GWA-X provides computational efficiency and flexibility for permutation testing in current and future GWAS.

As future work, GWA-X will be extended to support binary trait analysis, adaptive permutation, and ridge regression. In addition, regression models and permutation tests are used in various applications in omics research. Thus, another direction for future work is extending the regression method and the permutation test approach for different analyses, such as rare variant and single-cell RNA sequencing analyses.

8 Acknowledgments

This work used the ARCHER2 UK National Supercomputing Service (https://www.archer2.ac.uk). The work is funded by the Biotechnology and Biological Sciences Research Council (BBS/E/D/10002070, BBS/E/D/30002275, BBS/E/RL/230001A), Health Data Research UK (HDR-9003, HDR-9004), the Medical Research Council (MC_UU_00007/10) and the National Institute for Health Research (NIHR202639).



(a) False positive rate (FPR)

(b) Statistical power

Figure 11: The empirical (a) false positive rate and (b) power results of permutation-based and Bonferroni thresholds on skewed traits. Each box plot represents the distribution of the estimated quantity across 10 simulation replicates.

References

[1] Backman, J. D., Li, A. H., Marcketta, A., Sun, D., Mbatchou, J., Kessler, M. D., Benner, C., Liu, D., Locke, A. E., Balasubramanian, S., et al. Exome sequencing and analysis of 454,787 uk biobank participants. *Nature 599*, 7886 (2021), 628–634.

[2] Biobank, M. B. A. L. . W. S. J. . W. V. A. . M. J. G. . C. M. S. . ., and of Us Research Demonstration Project Teams Choi Seung Hoan 14 http://orcid. org/0000-0002-0322-8970 Wang Xin 14 http://orcid. org/0000 0001-6042-4487 Rosenthal Elisabeth A. 15, A. Genomic data in the all of us research program. *Nature 627*, 8003 (2024), 340–346.

[3] Mbatchou, J., Barnard, L., Backman, J., Marcketta, A., Kosmicki, J. A., Ziyatdinov, A., Benner, C., O'Dushlaine, C., Barber, M., Boutkov, B., et al. Computationally efficient whole-genome regression for quantitative and binary traits. *Nature genetics 53*, 7 (2021), 1097–1103.

[4] Jiang, L., Zheng, Z., Fang, H., and Yang, J. A generalized linear mixed model association tool for biobank-scale data. *Nature genetics 53*, 11 (2021), 1616–1621.

[5] Uffelmann, E., Huang, Q. Q., Munung, N. S., De Vries, J., Okada, Y., Martin, A. R., Martin, H. C., Lappalainen, T., and Posthuma, D. Genome-wide association studies. *Nature Reviews Methods Primers 1*, 1 (2021), 59.

[6] Pulit, S. L., de With, S. A., and de Bakker, P. I. Resetting the bar: Statistical significance in whole-genome sequencing-based association studies of global populations. *Genetic epidemiology 41*, 2 (2017), 145–151.

[7] Neale, B. M., Rivas, M. A., Voight, B. F., Altshuler, D., Devlin, B., Orho-Melander, M., Kathiresan, S., Purcell, S. M., Roeder, K., and Daly, M. J. Testing for an unusual distribution of rare variants. *PLoS genetics 7*, 3 (2011), e1001322.

[8] Chen, W., Coombes, B. J., and Larson, N. B. Recent advances and challenges of rare variant association analysis in the biobank sequencing era. *Frontiers in genetics 13* (2022), 1014947.

[9] Pahl, R., and Schäfer, H. Permory: an ld-exploiting permutation test algorithm for powerful genome-wide association testing. *Bioinformatics 26*, 17 (2010), 2093–2100.

[10] John, M., Ankenbrand, M. J., Artmann, C., Freudenthal, J. A., Korte, A., and Grimm, D. G. Efficient permutation-based genome-wide association studies for normal and skewed phenotypic distributions. *Bioinformatics 38*, Supplement_2 (2022), ii5–ii12.

[11] Joo, J. W. J., Hormozdiari, F., Han, B., and Eskin, E. Multiple testing correction in linear mixed models. *Genome biology 17* (2016), 1–18.

[12] Wood, A., Denholm, R., Hollings, S., Cooper, J., Ip, S., Walker, V., Denaxas, S., Akbari, A., Banerjee, A., Whiteley, W., et al. Linked electronic health records for research on a nationwide cohort of more than 54 million people in england: data resource. *bmj 373* (2021).

[13] Dean, J., Patterson, D., and Young, C. A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro 38*, 2 (2018), 21–29.

[14] Villa, O., Johnson, D. R., Oconnor, M., Bolotin, E., Nellans, D., Luitjens, J., Sakharnykh, N., Wang, P., Micikevicius, P., Scudiero, A., et al. Scaling the power wall: a path to exascale. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2014), IEEE, pp. 830–841.

[15] Reed, D., Gannon, D., and Dongarra, J. Reinventing high performance computing: challenges and opportunities. *arXiv preprint arXiv:2203.02544* (2022).

[16] Talib, M. A., Majzoub, S., Nasir, Q., and Jamal, D. A systematic literature review on hardware implementation of artificial intelligence algorithms. *The Journal of Supercomputing 77*, 2 (2021), 1897–1938.

[17] Mittal, S., and Vetter, J. S. A survey of cpu-gpu heterogeneous computing techniques. *ACM Computing Surveys (CSUR) 47*, 4 (2015), 1–35.

[18] Dongarra, J., Gates, M., Haidar, A., Kurzak, J., Luszczek, P., Tomov, S., and Yamazaki, I. Accelerating numerical dense linear algebra calculations with gpus. *Numerical Computations with GPUs* (2014), 1–26.

[19] Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (2024), pp. 929–947.

[20] Dongarra, J., Hammarling, S., Higham, N. J., Relton, S. D., Valero-Lara, P., and Zounon, M. The design and performance of batched blas on modern high-performance computing systems. *Procedia Computer Science 108* (2017), 495–504.

[21] CHE, R., JACK, J. R., MOTSINGER-REIF, A. A., AND BROWN, C. C. An adaptive permutation approach for genome-wide association study: evaluation and recommendations for use. *BioData mining 7* (2014), 1–13.

[22] RAJABLI, F., AND KUNKLE, B. W. Strategies in aggregation tests for rare variants. *Current protocols 3*, 11 (2023), e931.

[23] GURINOVICH, A., LI, M., LESHCHYK, A., BAE, H., SONG, Z., ARBEEV, K. G., NYGAARD, M., FEITOSA, M. F., PERLS, T. T., AND SEBASTIANI, P. Evaluation of genesis, saige, regenie and fastgwa-glmm for genome-wide association studies of binary traits in correlated data. *Frontiers in Genetics 13* (2022), 897210.

[24] CONSORTIUM, U. B. W.-G. S., LI, S., CARSS, K. J., HALLDORSSON, B. V., AND CORTES, A. Whole-genome sequencing of half-a-million uk biobank participants. *medRxiv* (2023), 2023–12.

[25] ABDELFATTAH, A., BEAMS, N., CARSON, R., GHYSELS, P., KOLEV, T., STITT, T., VARGAS, A., TOMOV, S., AND DONGARRA, J. Magma: Enabling exascale performance with accelerated blas and lapack for diverse gpu architectures. *The International Journal of High Performance Computing Applications* (2024), 10943420241261960.

[26] STEISS, V., LETSCHERT, T., SCHÄFER, H., AND PAHL, R. Permory-mpi: a program for high-speed parallel permutation testing in genome-wide association studies. *Bioinformatics 28*, 8 (2012), 1168–1169.

[27] PAIK, H., CHO, Y., CHO, S. B., AND KWON, O.-K. Mpi-gwas: a supercomputing-aided permutation approach for genome-wide association studies. *Genomics & Informatics 20*, 1 (2022).

[28] CHANG, C. C., CHOW, C. C., TELLIER, L. C., VATTIKUTI, S., PURCELL, S. M., AND LEE, J. J. Second-generation plink: rising to the challenge of larger and richer datasets. *Gigascience 4*, 1 (2015), s13742–015.

[29] FREUDENTHAL, J. A., ANKENBRAND, M. J., GRIMM, D. G., AND KORTE, A. Gwas-flow: A gpu accelerated framework for efficient permutation based genome-wide association studies. *BioRxiv* (2019), 783100.

[30] BI, W., ZHOU, W., DEY, R., MUKHERJEE, B., SAMPSON, J. N., AND LEE, S. Efficient mixed model approach for large-scale genome-wide association studies of ordinal categorical phenotypes. *The American Journal of Human Genetics 108*, 5 (2021), 825–839.

[31] SHEN, X., ALAM, M., FIKSE, F., AND RÖNNEGÅRD, L. A novel generalized ridge regression method for quantitative genetics. *Genetics 193*, 4 (2013), 1255–1268.

[32] HIJMA, P., HELDENS, S., SCLOCCO, A., VAN WERKHOVEN, B., AND BAL, H. E. Optimization techniques for gpu programming. *ACM Computing Surveys 55*, 11 (2023), 1–81.

[33] SHARMA, G., AGARWALA, A., AND BHATTACHARYA, B. A fast parallel gauss jordan algorithm for matrix inversion using cuda. *Computers & Structures 128* (2013), 31–37.

[34] DOMÍNGUEZ CONDE, C., XU, C., JARVIS, L., RAINBOW, D., WELLS, S., GOMES, T., HOWLETT, S., SUCHANEK, O., POLANSKI, K., KING, H., ET AL. Cross-tissue immune cell analysis reveals tissue-specific features in humans. *Science 376*, 6594 (2022), eabl5197.

[35] LAM, S. K., PITROU, A., AND SEIBERT, S. Numba: a llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (New York, NY, USA, 2015), LLVM '15, Association for Computing Machinery.

[36] SU, Z., MARCHINI, J., AND DONNELLY, P. Hapgen2: simulation of multiple disease snps. *Bioinformatics 27*, 16 (2011), 2304–2305.

[37] ARCHER2. Hardware - ARCHER2 user documentation, 2024. https://https://docs.archer2.ac.uk/user-guide/hardware [Accessed: 13 Sept 2024].

[38] COURTY, B., SCHMIDT, V., LUCCIONI, S., GOYAL-KAMAL, MARIONCOUTAREL, FELD, B., LECOURT, J., LIAMCONNELL, SABONI, A., INIMAZ, SUPATOMIC, LÉVAL, M., BLANCHE, L., CRUVEILLER, A., OUMINASARA, ZHAO, F., JOSHI, A., BOGROFF, A., DE LAVOREILLE, H., LASKARIS, N., ABATI, E., BLANK, D., WANG, Z., CATOVIC, A., ALENCON, M., STĘCHŁY, M., BAUER, C., DE ARAÚJO, L. O. N., JPW, AND MINERVABOOKS. mlco2/codecarbon: v2.4.1, May 2024.