Attacking the Opioid Epidemic: Determining the Epistatic and Pleiotropic Genetic Architectures for Chronic Pain and Opioid Addiction

Wayne Joubert[1], Deborah Weighill[1,4], David Kainer[1],
Sharlee Climer[2], Amy Justice[3], Kjiersten Fagnan[5], Daniel Jacobson[1]
[1]Oak Ridge National Laboratory, [2]University of Missouri-St. Louis,
[3]Yale University/Department of Veterans Affairs, [4]University of Tennessee, [5]DOE Joint Genome Institute

***Abstract*—We describe the CoMet application for large-scale epistatic Genome-Wide Association Studies (eGWAS) and pleiotropy studies. High performance is attained by transforming the underlying vector comparison methods into highly performant generalized distributed dense linear algebra operations. The 2-way and 3-way Proportional Similarity metric and Custom Correlation Coefficient are implemented using native or adapted GEMM kernels optimized for GPU architectures. By aggressive overlapping of communications, transfers and computations, high efficiency with respect to single GPU kernel performance is maintained up to the full Titan and Summit systems. Nearly 300 quadrillion element comparisons per second and over 2.3 mixed precision ExaOps are reached on Summit by use of Tensor Core hardware on the Nvidia Volta GPUs. Performance is four to five orders of magnitude beyond comparable state of the art. CoMet is currently being used in projects ranging from bioenergy to clinical genomics, including for the genetics of chronic pain and opioid addiction.**

I. Justification for ACM Gordon Bell Prize

We formulate vector similarity methods as generalized distributed dense linear algebra operations. Fast hardware features such as floating point minimum, population count and reduced precision arithmetic are exploited on Nvidia GPUs; the methods are also implementable on other modern processors with these features. Up to ***98% weak scaling efficiency*** is achieved at full system relative to single GPU kernel performance due to aggressive overlap of the expensive all-to-all communications. PS method performance reaches up to ***189 Petaflops*** single precision. CCC method operation rates are up to ***2.36 ExaOps*** on 99% of Summit, the ***first reported ExaOp calculation*** by a production application, achieving over 10^{18} mixed precision floating point operations per second. Performance is ***four to five orders of magnitude*** beyond best current state of the art. The methods are applicable more generally to vector similarity, distributed dense linear algebra and distributed tensor computations.

TABLE I: Performance attributes

Performance attribute	Value
Category of achievement	scalability, peak performance, time to solution
Type of method	dense vector similarity calculation
Results reported for	whole application including I/O
Precision reported	single, double
Precision used	half, single, double, multibit integer
System scale	Titan (full); Summit (full)
Measurement mechanism	manual timers, operation counters; NVPROF

II. Overview of the Problem

A. *Scientific Use Case: Chronic Pain and Opioid Addiction*

Opioids epidemic. Opioid misuse and addiction is having dramatic impacts on public health and social and economic welfare [1], [2], [3], [4]. The CDC has estimated that the economic burden of prescription opioid misuse alone is $78.5 billion per year in the United States [5]. Almost 30% of patients that are prescribed opioids misuse them [6], with over 10% developing an opioid use disorder [7], [8], [9]. As many as 6% of patients who misuse opioids will go on to use heroin [7], [8], [9]. There has been a 30% increase in opioid overdoses from July 2016 through September 2017 in 52 areas in 45 states [10]. In addition, there has been an increase in neonatal abstinence syndrome as opioid use and misuse during pregnancy has increased.

Over 50% of veterans suffer from chronic pain (compared to 30% in the general population). In a recent analysis among current veterans in care nationally, the Veterans Aging Cohort Study (VACS) found that 22.7% had sustained use of prescription opioids, which is substantially in excess of the rates in the general U.S. population. 68,000 veterans currently suffer from opioid use disorders. In prior work with the VA population, we found that opioid use was associated with negative health outcomes [11].

As such, opioid misuse has become a national (and Department of Veterans Affairs (VA)) crisis in which more than 115 people die each day in the United States due to opioid overdoses. Understanding the genetic architecture for how individuals develop chronic pain and respond to opioids and

SC18, November 11-16, 2018, Dallas, Texas, USA
978-1-5386-8384-2/18/$31.00 ©2018 IEEE

the identification of new therapeutic targets for opioid misuse would have a major impact on this growing epidemic. Chronic pain and opioid addiction are highly complex disorders. In contrast to Mendelian traits that arise due to a single mutation, chronic pain and opioid addiction are likely manifested due to the combined actions of multiple genetic factors that may be interacting additively or epistatically (i.e., combinations of genome variants that collectively lead to disease).

Epistatic genetic architectures for complex phenotypes. Genome Wide Association Studies (GWAS) and Quantitative Trait Loci (QTL) efforts seek to identify genetic variants that contribute to individual phenotypes, including susceptibility (or robustness) to disease. With the exception of Mendelian disease, risk alleles function in complex networks that are often challenging to dissect with extant statistical procedures, particularly when the effects of multiple variants are non-additive, or epistatic (for a recent review see [12]). Recent studies have made it clear that risks for many complex human diseases derive from non-additive interactions between multiple genes ([13]; for a review see [14])—including type I diabetes [15], coronary artery disease [16], and many others (for a review see [17]). If most non-additive effects in human disease are of higher order (e.g., three or more QTLs acting together), then we would not be aware of it, since existing statistical methods are incapable of mining such interactions from available sample sizes and limited computational resources. Epistatic networks have also been shown to drive susceptibility to complex diseases in genetic model organisms: studies in tractable model organisms have demonstrated that strong epistatic dependencies are the rule, rather than the exception, in higher metazoans [18], [19].

Thus, a grand challenge for population genetics is the construction of statistical and computational procedures that can identify epistatic networks at, or near, the same efficiency as single QTLs or risk alleles. However, the potential interactions in a human cell are on the order of 10^{170}, and thus exhaustive exploration of all possible molecular interactions has never been possible before. ***As described below, our network-based approach will allow us to build sets of SNPs representing higher-order combinations that would never be tested by existing methods.*** The computational methods presented in this work will allow us to reveal connections never previously discoverable.

Thus, the combination of algorithm development and leadership class supercomputing facilities allows us to break the longstanding curse of dimensionality that has historically plagued the study of complex biological systems. Brute force methods currently exist to perform epistatic GWAS [20]; however, they are not computationally tractable as it would take more than 10^{20} CPU hours to test all possible 4th order interactions of a single human phenotype. ***As such, the issue is not the ability to test sets of genome variants for association with a phenotype but rather what specific sets to test. Once the sets have been defined, set-based epistatic GWAS methods are completely computationally tractable [21], [22], [23]***. For example, the SKAT method [23] only takes 277 CPU hours on Titan to test 10 million SNP sets. Thus, our approach to this is to use the inherent co-evolutionary information resident in population-scale genome datasets themselves in order to determine a tractable number of sets to test for epistatic associations with chronic pain and opioid addiction phenotypes.

Custom Correlation Coefficient. Computation of the mathematical relationships between pairs of vectors is required in many science domains. In the field of genomics, the Custom Correlation Coefficient (CCC) [24] was developed to calculate the correlation between mutations (Single Nucleotide Polymorphisms (SNPs)) across a population of individuals. This can be used to identify groups of SNPs which tend to co-occur in a population, and consequently can be used to find combinations of SNPs which associate with certain phenotypes, such as a disease phenotype [21]. CCC also takes into account genetic heterogeneity and finds correlations between SNPs which co-occur in portions of the population, not requiring co-occurrence across the whole population. SNP correlations that pass a significance threshold can be modelled as a genome-wide network and interpreted as a co-evolutionary signal across the population with the hypothesis that genes involved in common functions/protein complexes are under selective pressure to co-evolve with one another. As such, connections between SNPs in these networks are ideally suited for epistatic set creation.

Furthermore, in order to capture more complex relationships, we have previously introduced a new ternary network definition, namely 3-way networks based on the concept of hypergraphs [25]. We define 3-way networks as network models of ternary relationships, i.e., relationships between triplets of objects. 3-way networks are defined by replacing the previous definition of an edge as a set of two nodes by a set of three nodes. Thus a 3-way network is a type of hypergraph [26]. We have previously shown that 3-way networks could discover more sophisticated patterns than are found by pairwise networks [25], [27]. We have recently created a 3-way version of the CCC algorithm and are using it in a complimentary fashion with the 2-way version.

SNP set creation. Breadth-first searches will be used on this network in order to determine the nearest neighbor of each and every node in the SNP correlation network, the resulting sets from each breadth-first search can be viewed as putative epistatic relationships and can be used for set-based genome-wide association testing against phenotypes. Furthermore, we have previously shown that principal component analysis of the Laplacian transform of a network adjacency matrix is an excellent method to extract topologies centered on every node in a network that are distinct from those topologies found by breadth first searches [28]. Finally, Markov clustering [29] will also be used to find topologies in the network to be used for set creation. ***The resulting sets extracted from the CCC network topologies represent higher-order combinatorial sets that allow for much more complex interactions to be tested than simple SNP pairs tested by extant methods.***

Ultimate application. Our close collaboration with the

Million Veteran Project (MVP) offers an unprecedented opportunity to understand links between human genetics, life history and propensity for chronic pain and opioid addiction. The sheer size of the MVP cohort and prevalence of chronic pain and opioid use and dependency in the VA population offers statistical power that has never been available before. The construction of human systems biology models combined with massive scale small molecule docking and molecular dynamics constitutes a new way forward for better molecular understanding and development of possible solutions for chronic pain and opioid addiction. Thus, SNP sets created via CCC networks as described above will be used to create sets of potentially epistatically interacting SNPs that can be tested for association with opioid addiction and chronic pain phenotypes that will, in close collaboration with the VA, be extracted from the electronic health records and patient genomic information to be made available for this study.

In addition, we are using model organisms such as *Drosophila*, mice and *Daphnia* in order to understand pain and opioid addiction phenotypes. Given that that each species will require on the order of 10^{16} SNP correlations to be calculated and subsequently thresholded this requires an incredibly efficient algorithm and an enormous amount of compute power to achieve.

The GPU optimized CCC algorithm we have developed for this work can meet the needs of these types of ambitious projects. Thus, pipelines we have constructed will enable the high-throughput discovery of epistatic networks directly from population genomics and phenomics data, providing a new tool for GWAS and QTL studies. The potential impact is substantial as we will resolve a fundamental question in genetics: we will learn the extent to which epistatic interactions drive emergent, complex phenotypes in opioid addiction. Once created, these epistatic networks and SNP sets can be used to test for associations with any phenotype that has been measured across a population. As such, this will allow for a tremendous amount of new information to be derived from publicly available phenotypes from a range of model organisms and the exploration of all of the clinical phenotypes (including alcohol and nicotine dependency, cardiovascular disease, prostate cancer and suicide ideation) that can be derived over time from the VA electronic health records as well as other such projects (NIH dbGAP, NIH All of Us Research Program, UK Biobank, etc) as data becomes available. Furthermore, this same approach can be used in other organisms of bioenergy or environmental interest that are being sequenced at population scale, such as those sequenced by the DOE Joint Genome Institute (*Populus trichcocarpa, Daphnia, Panicum, Brachypodium, Sphagnum, Sorghum, etc.*) as well as food crops (such as maize, soybean, wheat, etc.) supported by other funding agencies.

Proportional Similarity metric for pleiotropy discovery. Our overall research project in each of the species that we are studying involves the creation of extensive systems biology models of each species that capture the molecular interactions in the cell that lead to emergent properties and complex, organismal-scale phenotypes. This goal includes the determination of genome-variant to phenotype relationships for all available phenotypes. For some species we already have collected thousands of phenotypes (160,000 phenotypes in the case of our bioenergy project with *Populus trichocarpa*). The VA electronic health record provides a rich resource of potential clinical phenotypes and omics-level phenotype collection is currently underway. As such, pleiotropy is an important factor that needs to be captured in these extensive models. Pleiotropy is the phenomenon in which a gene is involved in multiple phenotypes. From a therapeutic point of view pleiotropy is very important to understand. If one identifies a gene as a therapeutic target (for chronic pain or addiction therapy), then it would be incredibly helpful to know what other functions, if any, that gene is involved in. This would allow us to have a predictive (rather than reactive) view on side affects and could greatly inform the choice of genes targeted for therapeutic intervention.

The discovery of pleiotropy can be reduced to a vector comparison problem. A matrix is first created in which the rows are SNPs and the columns are phenotypes. A "1" in the SNP vector represents a significant association between that SNP and a particular phenotype where a "0" represents the lack of a significant association with the phenotype in that column. Alternatively, a negative log transformation of the GWAS p-value can be used as a quantitative representation of the strength of the association.

All against all (either 2-way or 3-way) comparisons of SNP vectors is then performed with the Proportional Similarity metric described below. SNP-SNP comparison scores that are above a significance threshold indicate SNPs that are involved in similar sets of phenotypes. These relationships can be modeled as a network and Markov clustering [29] applied to produce sets of SNPs (pleiotropy modules) that are associated with the same set of phenotypes. These pleiotropy modules can be represented as a node in a tripartite network with edges to nodes representing the genes in which they reside and to nodes representing the phenotypes that they are associated with. This network then serves as a model of the pleiotropic patterns observed in that species and can be mined for patterns of interest and inspected when therapeutic targets are being selected.

Other uses of the Proportional Similarity metric. The Proportional Similarity metric can be used to compare binary or quantitative vectors for the degree of overlap/similarity. We have used this in the past to compare the gene family content of species and to thus reveal functional relationships and similarities between species [25]. Similarly it can be used on the transpose of such vectors to reveal co-occurrence and co-expansion patterns among gene families to infer functional co-evolution patterns. We are working on a project now that will enable these types of comparisons across all publicly available genomes (>120,000 genomes and thus millions of gene families). The Proportional Similarity metric can also be used to determine microbial community structure in microbiome studies. In addition, it has many applications outside the field of biology as it compares any set of numeric vectors

for their degree of overlap. Whether it is being used for pleiotropy discovery or these other purposes, the scale of the computation of comparison of millions of vectors certainly requires efficient, scalable performance on supercomputing architectures as demonstrated later in this paper.

B. Methods

As discussed above, we consider two methods: the Proportional Similarity (PS) metric [27] and the Custom Correlation Coefficient [24], [30]. The (2-way) Proportional Similarity metric for vectors u and v with real nonnegative entries is

$$c_2(u,v) = 2\Big[\sum_q min(u_q, v_q)\Big] \Big/ \Big[\sum_q (u_q + v_q)\Big] \quad (1)$$

where $min(\cdot,\cdot)$ takes the minimum of two scalars. For a set of vectors, the metrics for all pairs form a square matrix with roughly half of the entries unique, due to symmetry of $c_2(\cdot,\cdot)$.

To measure similarity of three vectors u, v and w, the 3-way Proportional Similarity metric is

$$c_3(u,v,w) = (3/2)\Big[\sum_q min(u_q, v_q) + min(u_q, w_q) + min(v_q, w_q) - min(u_q, v_q, w_q)\Big] \Big/ \sum_q (u_q + v_q + w_q). \quad (2)$$

For a set of vectors, the metric values for all vector triples form a 3D tensor roughly 1/6 of whose entries are unique, due to symmetry of $c_3(\cdot,\cdot,\cdot)$.

The PS metric can be computed in single or double precision, referred to here as PS/SP and PS/DP. Some cases only require 1-2 digits of accuracy for each result; in such cases, if vector length is not too large, single precision is adequate.

The CCC method assumes vectors whose elements each consist of two bits, directly encoding the SNP allele data (see [24], [30]). The chief computation of the 2-way and 3-way methods is, for each pair or triple of vectors, to generate a tally table containing sums of occurrences of the relevant bit patterns in corresponding vector elements. For details see Appendix I. As with the PS method, when comparing all vectors in a set, only about 1/2 (1/6) of the 2-way (3-way) values are unique.

In practice, genomic data may contain missing entries. For the PS metric, a missing entry can be represented as a zero vector element with no change to the method. However, this approach is not viable for CCC, for which zero entries are signficant. Following [30], an altered method is defined which encodes the missing value as a special 2-bit vector entry and skips the tally accordingly. This method is denoted CCC/sp.

We have adapted the CCC methods to use the Tensor Core hardware of the Nvidia Volta GPUs (see Appendix I). We denote these implementations CCC/tc and CCC/sp/tc.

The datasets we have used thus far in production contain no more than 5-10% missing entries. Thus there is no advantage to treating the input data as a sparse rather than dense matrix.

III. Current State of the Art

Vector similarity methods are frequently used for detection of complex traits through GWAS; for a survey see [31]. The increase in genomic data in recent years has sparked growing interest in accelerating these methods. Previous efforts have adapted these calculations to GPUs or Intel Xeon Phi processors; a smaller number of efforts have parallelized these methods on large-scale HPC systems. We survey these here, considering cases for which vectors are composed of allele data in the form of 2-bit or 3-bit values (similarly to the CCC method), which is the predominant approach in the field. We note some efforts have used nonexhaustive search methods; since the PS and CCC methods considered here are exhaustive search methods, we do not compare to nonexhaustive search.

GBOOST [32] is a gene-gene interaction code for 2-way studies optimized for single GPUs using encoding of gene data into bit strings with avoidance of redundant computations. GWISFI [33] is a single-GPU code for 2-way GWAS calculations reporting 10,000X faster performance than the popular PLINK code. [34] develops a UPC++ code for gene-gene interaction studies for small numbers of GPUs and Intel Xeon Phi processors exploiting vector hardware and hardware population count instructions. [35] calculates 3-way interactions on a node with 4 GPUs. [36] considers k-way GWAS studies for arbitrary k with consideration of load balancing and elimination of redundancies and presents 2-way results on a 4096-node IBM Blue Gene/Q system; single GPU results are also presented. [37] performs 2-way analyses on up to 126 nodes of the Intel Phi-based Stampede system (cf. [38]). multiEpistSearch [39] performs 2-way GWAS studies using UPC++ on a 24 GPU cluster. [40] describes 2-way methods executed on 512 nodes of the Edison system.

Table II shows rates of comparison of element pairs (or triples) per second as reported in these references. The methods in some cases compute very different statistics but are all based on the same fundamental computation on vectors of alleles. Element comparison rates have been either taken directly as reported or calculated from the data presented therein. We also present results for CoMet, the comparative genomics application used in the present work. CoMet CCC results are directly comparable to the other studies; CoMet PS results are different in kind but are listed here for completeness.

On 4560 nodes (99.0%) of the ORNL Summit system, CoMet 2-way CCC/sp/tc is ***21,285X faster*** than [40], the fastest known comparable result.

The fastest 3-way implementation with reported performance numbers we are aware of is [35]. At 4373 Summit nodes, CoMet 3-way CCC/sp/tc is ***306,910X faster*** than [35].

The dramatic improvement achieved by CoMet over state of the art is due to coupling a highly efficient GPU implementation with a methodology to maintain efficiency across of tens of thousands of GPUs.

IV. Innovations Realized

Multiple innovations were developed to achieve the speedups shown in this work. Some have previously been reported in different contexts in the literature; to our knowledge, this is the first effort to bring all the elements together to achieve petascale and exascale vector similarity calculations for comparative genomics problems.

TABLE II: Comparison to related work

code	problem	node config	nodes used	cmp/sec ($\times 10^9$)
GBOOST[32]	2-way GWAS	1 Nvidia GTX 285	1	64.08
GWISFI[33]	2-way GWAS	1 Nvidia GTX 470	1	767
[36]	2-way GWAS	1 Nvidia GTX 470	1	649
[36]	2-way GWAS	IBM Blue Gene/Q	4096	2520
epiSNP[37]	2-way GWAS	2 Intel Phi SE10P	126	1593
[34]	2-way GWAS	2 Nvidia K20m + 1 Intel Phi 5110P	1	1053
multiEpistSearch [39]	2-way GWAS	1 Nvidia GTX/Titan	24	12,626
[40]	2-way GWAS	2 Intel Xeon E5-4603	512	13,889
CoMet, Titan	2-way PS/SP	1 Nvidia K20X	17472	4.289e6
CoMet, Titan	2-way PS/DP	1 Nvidia K20X	17472	1.697e6
CoMet, Titan	2-way CCC	1 Nvidia K20X	17955	9.108e6
CoMet, Summit	2-way PS/SP	6 Nvidia V100	4560	94.768e6
CoMet, Summit	2-way PS/DP	6 Nvidia V100	4560	29.586e6
CoMet, Summit	2-way CCC	6 Nvidia V100	4560	104.370e6
CoMet, Summit	2-way CCC/sp	6 Nvidia V100	4560	71.587e6
CoMet, Summit	2-way CCC/tc	6 Nvidia V100	4560	294.652e6
CoMet, Summit	2-way CCC/sp/tc	6 Nvidia V100	4560	295.633e6
GPU3SNP[35]	3-way GWAS	4 Nvidia GTX/Titan	1	264.7
CoMet, Titan	3-way PS/SP	1 Nvidia K20X	18424	5.695e6
CoMet, Titan	3-way PS/DP	1 Nvidia K20X	18424	2.445e6
CoMet, Titan	3-way CCC	1 Nvidia K20X	18424	2.058e6
CoMet, Summit	3-way PS/SP	6 Nvidia V100	4373	72.499e6
CoMet, Summit	3-way PS/DP	6 Nvidia V100	4373	27.755e6
CoMet, Summit	3-way CCC	6 Nvidia V100	4373	23.672e6
CoMet, Summit	3-way CCC/sp	6 Nvidia V100	4373	21.163e6
CoMet, Summit	3-way CCC/tc	6 Nvidia V100	4373	81.611e6
CoMet, Summit	3-way CCC/sp/tc	6 Nvidia V100	4373	81.239e6

A. 2-way methods recast as modified GEMM operations

Computing similarity metrics on n vectors of length m in general requires $O(n^2m)$ operations on $O(nm)$ data, suggesting that restructuring for high computational intensity may be possible. Indeed, previous efforts to map these calculation to GPUs have taken this direction. However, a more direct approach is to exploit the structural resemblance of vector similarity calculations to BLAS-3 dense linear algebra matrix-matrix product (GEMM) operations. In fact, cosine similarity can itself be computed using a GEMM. This connection has already been recognized in the context of chemical informatics methods [41]; to our knowledge, our work is its first application to genomics calculations. This approach makes it possible to exploit the many advances in highly optimized GEMM implementations. The specific approach used here is to adapt the GEMM operations of the MAGMA [42] library. For the PS methods, this requires replacing the `c += a*b` GEMM scalar operation with `c += min(a,b)`. For CCC, a more complex modification is required involving bit-level operations, (or, for the Tensor Core methods, a restructuring to enable use of a standard mixed precision GEMM). We refer to these as modified GEMM operations.

B. Use of high performance hardware features

To optimize use of resources, we exploit fast hardware instructions using CUDA intrinsics. For the PS methods, the CUDA `fminf` and `fmin` functions quickly take the minimum of two values. This is, to our knowledge, the first use of these intrinsics for vector similarity computations. Their use yielded highest performance of several approaches tested on Titan's K20X GPUs. On Summit's V100 GPUs, the single precision `fminf` was fastest for PS/SP, however for PS/DP the double precision `fmin` was outperformed by casting to floats and applying `fminf`. Though some precision may be lost, the result is accumulated in double precision, avoiding critical loss of significance effects for long vectors, which is the major issue. Note similar instructions are available on other processors, e.g., the Intel Xeon Phi AVX512 instructions `_mm512_min_ps` and `_mm512_min_pd`.

For the CCC methods we use the `__popcll` CUDA intrinsic to count the 1 bits in a word. Similar intrinsics are available on some modern CPUs, e.g., `_mm_popcnt_u64`. This approach is also used in efforts such as [41], [34].

For Summit's Volta V100 GPUs we make use of the `cublasGemmEx` function to increase performance of the CCC methods by exploiting the Tensor Core hardware. This is done by mapping CCC input vector elements to half precision FP16 numbers and applying a matrix-matrix product (see Appendix I). A similar technique could be implemented using reduced precision hardware on other processors.

C. 3-way methods via multiple modified GEMMs

The 3-way methods compute a cube-shaped 3D tensor of results. Our approach is to represent this as a sequence of modified GEMM operations each for a plane of the cube. This exploits the high performance of modified GEMMs for the 2-way case. This "BLAS-4"-like operation has a further advantage: the cube of results computed on a node for one step requires only node-local information, thus no off-node communication is required while computing the block of results. For the PS methods, the 3-way term $min(u_q, v_q, w_q)$ of Equation (2) is computed with one modified GEMM for each plane of results [27]. For CCC, a more complex method uses three modified GEMMs for each plane [30].

D. Removing redundant computations, achieving load balance

The ScaLAPACK project [43] showed that efficient distributed dense linear algebra often requires multidimensional parallelism. We implement three parallelism axes: partitioning the set of input vectors, decomposing the vectors along their length, and replicating the vectors to distribute computation of result blocks. This results in a 3D grid of MPI ranks, each owning one GPU and a set of CPU cores in a NUMA domain.

Some previous efforts to parallelize these computations broadcast a full copy of the input dataset to each compute node. This is not practical here, however. CCC applied to an anticipated dataset of 10 million SNPs for 4 million individuals gives input data size of 10 TB—300X the 32 GB node memory of Titan and 20X the 512 GB node memory of Summit. Thus the input data must be distributed across nodes.

Since an all-to-all comparison of vectors is performed, the communication requirement is substantial. The computation is scheduled as a series of steps in which each partition of vectors is compared against that owned by another MPI rank specified by an offset. Asynchronous pipelining hides communication

under computation. CPU/GPU transfers are scheduled asynchronously during GPU computations and pipelined. Finally, computations left on the CPU are performed concurrently with GPU computations and parallelized with OpenMP.

To avoid performing unnecessary computation of redundant values due to symmetries while maintaining load balance, the subset of values to compute must be chosen carefully. The criteria are: (1) all unique metric values should be computed at least once with as few repeated computations as possible; (2) the computation must be load balanced across MPI ranks; (3) the data on each rank should be selected so that the modified GEMM operations are as large as possible, for high performance—ideally, of the same dimension as the vector partition associated with the rank. The work of [44] addresses similar issues in the context of distributed tensor computations with symmetries but is not concerned with (3).

For 2-way methods, it is typical to take the upper triangular part of the results matrix as representative of the unique values. However, to maintain load balance we instead take a block circulant subset of values, based on the block decomposition imposed by the partitioning of vectors (Figure 1) [27]. In this scheme every rank owning a block row has nearly the same number of blocks, thus the computation is load balanced.

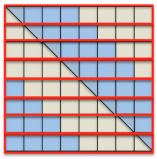

Fig. 1: Metric values computed for 2-way method (shaded). Block rows correspond to partitioning of vectors across ranks.

For the 3-way methods, the cube of results has an implied partitioning into subcubes imposed by the vector decomposition, with each MPI rank owning a 2D slab (Figure 2). One might consider six tetrahedral solid regions of the full cube of results, each with a separate a copy of unique results. However, any such tetrahedron intersected with the block partition does not give a load balanced result. To achieve load balance, our strategy is instead to select a subset of values from each subcube of Figure 2 so that every unique value of the whole cube is represented exactly once (Figure 3) [27]. For each subcube, a 1/6-width small slab of planes is selected. Its orientation and placement in the subcube is chosen so that, as can be seen from a folding argument around the cube's main axis, every subcube of any reference tetrahedron is fully populated. By decomposing the subcube into properly oriented planes, the modified GEMMs can operate on large matrices.

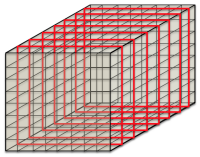

Fig. 2: 3-way method parallel decomposition.

The enormous quantity of metrics data created by the 2-way method and especially the 3-way method puts pressure

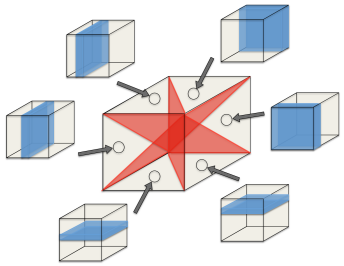

Fig. 3: Subset of elements selected from each subblock, depending on location in the domain.

on available node memory, severely limiting matrix size and thus efficiency of the GPU. To remedy this, for the 3-way case the small slab of planes in the subcube is decomposed into "stages"; in practice one stage at a time is computed, reducing storage needed for metrics at any given time. Similarly, for the 2-way and 3-way methods the computation is decomposed into "phases," each associated with a round-robin subset of the blocks in each block row (2-way) or slab (3-way). This limits main memory needed at any one time, so larger problems can be run and larger matrices given to the GPU.

V. How Performance Was Measured

A. Measurement methodology

Timings of major code components (each typically several seconds or more) are done manually using `gettimeofday` preceded by calls to `cudaDeviceSynchronize` and `MPI_Barrier` to quiesce the nodes. The breakdown of code components timed is described in the following section. The times for the job launch process via `aprun` (Titan) or `jsrun` (Summit) and for `MPI_Init` and `MPI_Finalize` are not included, as these are artifacts of system software performance and not germane to performance of the application proper. Performance data for GPU kernels is collected using NVPROF or the CUDA Profiler.

Following convention, a vector element comparison is defined for the 2-way methods as the computation pertaining to two corresponding elements of the two vectors compared. Likewise for the 3-way methods, an element comparison refers to three corresponding elements of the three vectors compared.

Redundant computations due to symmetries are never multiply counted: for example, comparing a to b and b to a counts as one comparison. Thus if redundant calculations are performed, time required for these computations appears in the analysis as a penalty to the achieved rate of operations per second.

For the PS method, the operations of floating point add and multiply are each counted as one operation, as is the operation of taking the minimum of two floating point values by intrinsic call to hardware instruction. When needed, a cast between floating point types is also counted as one operation. Operation counting is done manually for both numerators and denominators for the PS method; these counts can be computed analytically since, as analogues of dense linear algebra operations, their operation counts are well-understood. Any incidental operations that may be performed but not counted in this way will appear as a penalty against the operation rate.

For PS/SP, all computations counted are single precision; for PS/DP, all counted operations are double precision except for `fminf` and cast between floating point types on Summit.

For CCC, the operations counted are INT64 arithmetic, bitwise logical, shift and population count, floating point arithmetic and INT64 cast to double. These are counted by manual inspection of hand-tuned code. Most instructions executed are non-floating point bit manipulation operations. For the Tensor Core implementations, mixed precision scalar floating point multiply and add each count as one operation, and counts are calculated analytically by formula. Half precision add and multiply are here counted as arithmetic operations on floating point numbers. For both CCC and PS methods, instruction rates are also collected using NVPROF on Summit.

TABLE III: Titan GPU kernel results

Kernel	elt pairs / sec $\times 10^{12}$	ops / elt pair	ops / sec $\times 10^{12}$
PS/SP	0.495	2	0.990
MAGMA SGEMM	0.614	2	1.229
SGEMM limit	1.967	2	**3.933**
PS/DP	0.199	2	0.397
MAGMA DGEMM	0.308	2	0.617
DGEMM limit	0.656	2	**1.311**
CCC 2-way	0.583	60	1.093
CCC 3-way	0.408	82	1.045
CCC/sp 2/3-way	0.370	88	1.018
MAGMA ZGEMM	0.092	8	0.736
ZGEMM limit	0.164	8	**1.311**

TABLE IV: Summit GPU kernel results

Kernel	elt pairs / sec $\times 10^{12}$	ops / elt pair	ops / sec $\times 10^{12}$	IPC
PS/SP	5.060	2	10.120	3.430
MAGMA SGEMM	6.071	2	12.142	2.410
SGEMM limit	7.834	2	**15.667**	—
PS/DP	1.272	5	6.362	1.362
MAGMA DGEMM	3.201	2	6.401	1.436
DGEMM limit	3.917	2	**7.834**	—
CCC 2-way	3.931	60	7.371	1.351
CCC 3-way	3.014	82	7.722	1.315
CCC/sp 2/3-way	2.670	88	7.342	1.343
MAGMA ZGEMM	0.650	8	5.203	1.169
ZGEMM limit	0.979	8	**7.834**	—
CCC/tc 2/3-way non/sparse	13.111	8	104.885	—
cuBLAS FP16/32 GEMM	14.035	8	112.282	—
FP16/32 GEMM limit	15.668	8	**125.344**	—

In production we have used a *Populus trichocarpa* dataset composed of 28,342,758 SNPs with a population size of 882, the subject of study under a current DOE INCITE project on Titan. In the future we will apply the code to a larger dataset of 700,000 SNPs and population size 358,000; a follow-on effort will analyze a 10 million SNP, 4 million population dataset. To evaluate performance, the code is also able to calculate with synthetically generated datasets for which the exact solution is known and thus results can be easily validated. Since the code path executed for the core metrics computation is entirely independent of the values present in the input data, the timing results from the synthetic datasets would be expected to be fully reflective of those from equivalent sized real-world datasets; this has been confirmed through testing. The Tensor Cores on Summit do exhibit data-dependent performance due to power/frequency throttling, however we have determined that the highly randomized synthetic test inputs used here closely represent performance for real-world input data (or are in fact slightly more challenging). I/O performance does depend on the input data, since code outputs are filtered by size threshold, so the number of outputs depends on the values (see next section). Thus we do not measure I/O for synthetic datasets.

Correctness of computations is ensured in multiple ways. The CoMet application has unit testing and an extensive suite of tests. Also the synthetic datasets have known analytic solutions against which results are compared for each run. A result checksum independent of parallel decomposition is computed to detect any defect in the computation. Computed results have also been validated by several other codes, including an offline C code to recompute results passing the output threshold and a code independently implemented in R to check these values.

B. Systems used

Experiments are performed on the ORNL Titan Cray XK7 system and the ORNL IBM AC922 Summit system. Systems and execution environment are described in Appendix II.

VI. Performance Results

A. GPU kernel results

To provide a baseline for performance achievable at scale, timing experiments are performed for large matrices on one GPU. Tables III and IV show rates based on GPU kernel execution time for a modified GEMM for each method. We measure rate of comparisons between corresponding elements of two vectors being compared, each element being FP32, FP64 or a 2-bit value. The number of operations required for a single comparison is also shown, counted as described earlier. Note some operations could theoretically be elided due to compiler optimizations, resulting in different operation rates; we do not believe this is significant here. For Summit, the instructions per clock (IPC) figures reported by NVPROF are also shown. Observe that, depending on whether the instructions are prinarily 32-bit (PS/SP, SGEMM) or 64-bit (others), the IPC values are fairly consistent between the true GEMM methods and the modified GEMMs.

The PS operation rates for Summit are 65% (single) and 81% (double) of theoretical peak operation rates for SGEMM and DGEMM, respectively, these derived from the GPU peak flop rates. This is highly performant insofar as modified GEMM performance, relying on `fminf`, cast-to-float and cast-to-double, must compete against highly optimized fused multiply add (FMA) instructions used by the standard GEMMs. It is unfortunate that the fastest implementation of PS/DP requires two cast-to-float operations and a cast-to-double operation with the `fminf` operation; one would prefer better V100 `fmin` performance. The corresponding PS ratios for Titan are 25% (single) and 30% (double), showing the architectural improvements of V100 over K20X.

For the CCC cases, comparison to ZGEMM theoretical peak is tenuous since the operations performed are very

different; however, the similar operation rates between the CCC modified GEMMs and the ZGEMM limit are suggestive that highly effective use of the GPU is being made.

For the Summit Tensor Cores, a benchmark result is obtained using cuBLAS with FP16 inputs and FP32 results computed in FP32 arithmetic, using large matrices filled with zeros. Experiments have shown that Tensor Core performance on Summit is data dependent due to power/frequency throttling for high entropy inputs, therefore this value serves as a high upper limit for HGEMM. The CCC measurement shown using the Tensor Cores is computed using matrices derived from random CCC inputs and is expected to be a practical upper bound for realistic CCC problems (see Appendix I).

B. Runtime components

A single run of the CoMet application is composed of several components, for each of which timings are collected: the vector similarity calculation proper (we refer to this as the "core metrics computation" throughout), initialization and deallocation of data structures for vectors and metrics, input of vectors and output of metrics. To illustrate overheads of operations other than the core metrics computation, we present timings from a sample run performed on Summit with the preliminary AlpineTDS GPFS file system used for data input and the NVMe burst buffers for output. A publicly available human genome dataset with 81,042,272 SNPs of population size 2504 is used. To evaluate performance on larger-population datasets not yet cleared for use on Summit, this population is replicated by 240X to give 600,960 population size. Computation and I/O behaviors on this dataset will mimic those of larger-population unreplicated datasets to be run in the near future.

2-way CCC/sp/tc is run on 3000 nodes of Summit with 125 phases executed in a single run. The Tensor Core computation is run in 6 GEMM steps for each full GEMM operation (Appendix I). We use the typical output metric threshold of 0.7, so that about one of every 3.5 million metric results is output. The core metrics computation runs at 10.40e12 element comparisons per second per rank (cp. Table VII), or 83.2 TF per GPU, due to use of the Tensor Cores, with full rate of ***1.50 ExaOps on nearly 2/3 of Summit*** for this realistic problem. On full Summit we estimate performance in excess of 2.25 ExaOps, similar to maximum weak scaling performance shown below. Note this problem, requiring ***3.3 hours total wallclock runtime*** on 2/3 of Summit, if it were run at a rate equivalent to best comparable state of the art (Table II), would require ***15 years wallclock runtime*** to complete.

Timing results are shown in Table V. Input cost is amortized over the many phases of the computation and thus accounts for less than 6% of runtime. Output cost is amortized by the long vector length due to the large population size and is limited by thresholding, thus accounting for only about 5% of runtime. The large metrics arrays are expensive to allocate as pinned memory but are reused across phases, thus amortizing cost to less than 1%. Vector allocation costs are negligible. The core metrics computation itself is efficient due to the large vector length and large number of vectors on each GPU. As a result, about 89% of runtime is spent in the core metrics computation.

In the near future, run campaigns with vector lengths as high as 4,000,000 are planned. These large population sizes will cause the core metrics computation to dominate to an even greater extent. Since I/O and other costs constitute only a small fraction of runtime, for the remainder of this paper we focus solely on timings of the core metrics computation.

TABLE V: Timings in seconds for execution components

component	time (sec)	percent
core metrics computation	10,550.23	88.80
vectors initialization	0.24	0.00
metrics initialization	58.44	0.49
input	670.93	5.65
output	600.40	5.05
TOTAL	11,880.25	100.00

C. Strong scaling results

Though the primary focus of this work is weak scaling regimes to solve very large problems, we give representative strong scaling results. Figure 4 shows 2-way PS/DP and CCC/sp/tc strong scaling up to 4000 nodes of Summit for two fixed problem sizes. For each case, the calculation was performed as a loop over phases within a single code execution, to reduce memory pressure from the large metrics storage requirement and thus enable a larger problem to be run. PS/DP (CCC/sp/tc) reaches 48% (47%) parallel efficiency at 4000 nodes relative to 500 nodes. It is well-known that standard GEMM operations require large problem sizes on the GPU to reach high operation rates. Furthermore, transfer, communication and CPU computation costs become significant as modified GEMM sizes per GPU are reduced. To achieve high performance in production, we will run large cases that fill GPU memory as much as possible.

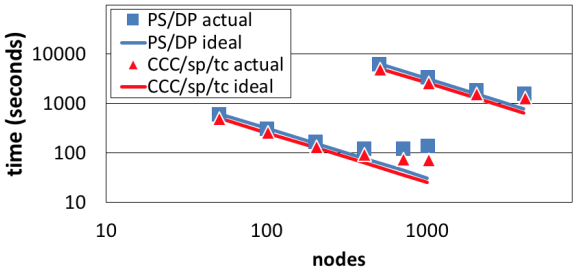

Fig. 4: 2-way PS/DP and CCC/sp/tc strong scaling on Summit

D. Weak scaling results

Figure 5 shows 2-way and 3-way weak scaling timing results on Titan and associated element comparison rates reported on a per rank basis. For further details, see [27], [30]. All cases scale well to the entire Titan system, with two caveats. First, the 2-way PS methods experience some drop in performance at large node counts. This is believed to be due to limitations of Titan's 3D torus interconnect for sending the large quantities of data; the other methods with higher computation to communication ratio do not manifest this behavior. Second, the 3-way methods at low node counts show

some reduction in comparison rates compared to large node counts. This is an artifact of the construction of the algorithm, which favors large node counts. However, by about 100 nodes, performance approaches the asymptote, and performance is high at the largest node counts.

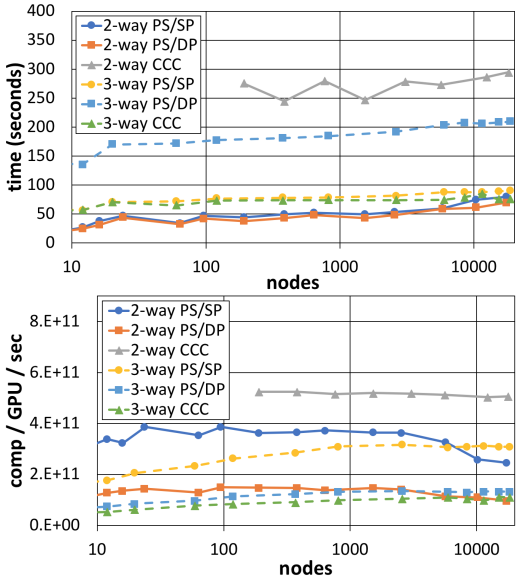

Fig. 5: Titan weak scaling timings and comparison rates

A summary of results at highest node counts is presented in Table VI. Performance per rank (i.e., per GPU) of the largest run is compared against the maximal measured GPU kernel performance from Table III; recall the 3-way CCC methods require three modified GEMMs per element comparison, while the PS methods require one. The percentage of maximal kernel performance attained by the core metrics computation is in the 49-87% range; omitting the lower performing 2-way PS methods, the range is 62-87%. This indicates all overhead from communication, CPU-GPU transfers, CPU computations and potential load imbalance is small compared to performance of the critical modified GEMM computations on the GPU.

TABLE VI: 2-way, 3-way methods performance on Titan

method	num way	nodes	cmp / sec all nodes $\times 10^{15}$	cmp / sec per GPU $\times 10^{12}$	max limit per GPU $\times 10^{12}$	ratio per GPU
PS/SP	2	17472	4.289	0.245	0.495	49.6%
PS/DP	2	17472	1.697	0.097	0.199	48.8%
CCC	2	17955	9.108	0.507	0.583	87.0%
PS/SP	3	18424	5.695	0.309	0.495	62.4%
PS/DP	3	18424	2.445	0.133	0.199	66.7%
CCC	3	18424	2.058	0.122	0.136	82.1%

Figure 6 shows weak scaling results on Summit. For 2-way cases, several phases are computed and phase count or the processor replication factor is increased with node count to keep work per node nearly constant. For 3-way cases, one stage of a fixed number of stages is computed, and the processor count along the parallel replication axis is increased with problem size to keep work per GPU nearly constant. Timings for all methods exhibit near-perfect weak scaling behavior up to 4560 (2-way case) or 4373 (3-way case) nodes, representing 99.0% (94.9%) of Summit. Summit's fat tree interconnect with adaptive routing effectively manages the large volume of matrix data transfer. There is slight performance loss at the highest node counts for 2-way PS/SP and 2-way Tensor Core methods, having lower computational intensity; this performance may improve in the future with further tuning of the network. Figure 6 shows element comparison rate per GPU also scales nearly perfectly. As with Titan, the 3-way methods have lower comparison rates at low node counts but have high performance for more nodes.

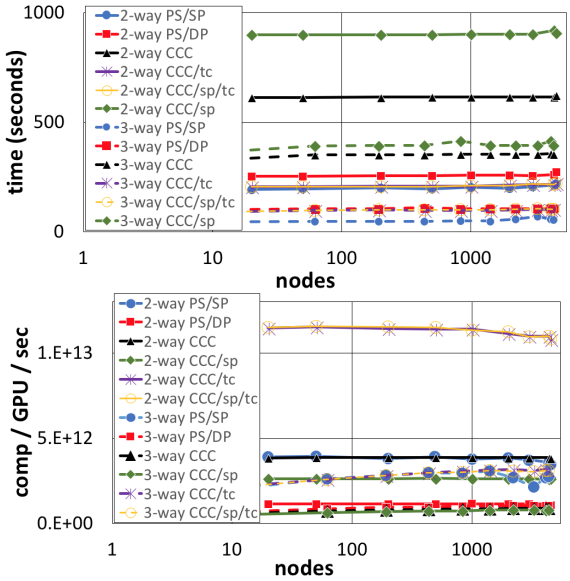

Fig. 6: Summit weak scaling timings and comparison rates

Table VII shows maximal performance. For the 2-way case, at 4560 nodes the core metrics computation attains 68-98% of maximum measured performance of the GPU kernel shown in Table IV. Note CCC and CCC/sp without Tensor Cores achieve near perfect efficiency due to high computational intensity and deep communication/transfer pipeline. For the 3-way case, at the largest node count the percentages are 55-90%; the ratio is over 70% for all CCC methods. The code makes very efficient use of the GPUs at large node counts.

TABLE VII: 2-way, 3-way methods performance on Summit

method	num way	nodes	cmp / sec all nodes $\times 10^{15}$	cmp / sec per GPU $\times 10^{12}$	max limit per GPU $\times 10^{12}$	ratio per GPU
PS/SP	2	4560	94.768	3.464	5.060	68.5%
PS/DP	2	4560	29.586	1.081	1.272	85.0%
CCC	2	4560	104.370	3.815	3.931	97.0%
CCC/sp	2	4560	71.587	2.616	2.670	98.0%
CCC/tc	2	4560	294.652	10.770	13.111	82.1%
CCC/sp/tc	2	4560	295.633	10.805	13.111	82.4%
PS/SP	3	4373	72.499	2.763	5.060	54.6%
PS/DP	3	4373	27.755	1.058	1.272	83.1%
CCC	3	4373	23.672	0.902	1.005	89.8%
CCC/sp	3	4373	21.163	0.807	1.005	80.3%
CCC/tc	3	4373	81.611	3.111	4.370	71.2%
CCC/sp/tc	3	4373	81.239	3.097	4.370	70.9%

2-way PS/SP achieves 94.768 petacomparisons per second at 4560 nodes; since PS/SP uses 2 operations per comparison (`fminf`, add), this represents ***189.54 PetaOps*** single precision.

2-way PS/DP reaches 29.586 petacomparisons per second; since PS/DP uses 5 operations per comparison (cast to float (2), `fminf`, cast to double, add), this yields ***147.93 PetaOps***. 2-way CCC/sp/tc reaches ***295.633 petacomparisons per second***; since each comparison uses 4 multiplies and 4 adds, this reaches ***2.365 ExaOps***. 2-way CCC/sp/tc computes the same result at ***4.130X*** higher performance compared to CCC/sp.

VII. Implications

We have presented a novel reformulation of vector similarity calculations in terms of generalized BLAS-3 operations. This effort and others, for example the Exascale Computing Project's Center for Efficient Exascale Discretizations (CEED), are needed to convert memory-bound computations into compute intensive operations. The power cost for moving data is increasingly a critical performance limiter as manifested by the growing trend toward memory-centric designs. Application codes will increasingly need to adapt to keep pace with this shift.

For this work, the treatment of the on-node problem as a modified GEMM will remain effective as long as future HPC hardware supports dense linear algebra—this is likely in view of continued application requirements for this [45]. Growing availability of on-chip silicon for specialized operations like `fmin` and `__popcll` has served this work well and might also benefit other algorithmic patterns. Reduced precision processor functionality seems likely for the near future owing to its value for the growing deep learning market.

Other efforts are underway to use Volta Tensor Cores for purposes other than deep learning [46]. It is hoped that more applications will be able to use this feature. The Titan CAAR application readiness project [47] examined use of reduced precision for the six targeted applications; only one could use mixed precision (LAMMPS), and in general few science applications have used reduced precision effectively. However, the high speedup factor of the Tensor Cores over double precision may suggest new possibilities.

Node heterogeneity and complexity will only increase [48]. As the slowing of Moore's law causes processor design to become more exotic, application developers will need to think of new method and algorithm formulations to take advantage of this hardware to advance their science goals.

The approach used here is not specific to GPUs but could be used for other advanced processors as well. Many modern CPUs provide a hardware population count instruction. Furthermore, just as this work modifies MAGMA BLAS functions, it is likely that other libraries, e.g., PLASMA [49], BLIS [50] and OpenBLAS [51] would provide similar opportunities for adapting the modified GEMM operation to conventional processors or Intel Xeon Phi.

The slower growth of off-node communication bandwidth than node flop rates on HPC systems is a challenge for distributed dense linear algebra in general and these methods in particular. Communication requirements are much greater than simple halo exchanges required by many codes. In this study, some matrix message sizes were on the order of a gigabyte; the flood of large messages poses a threat of significant network contention. The 2-way PS and Tensor Core methods are most vulnerable to this hardware stress point; the 3-way methods have a BLAS-4 character with higher achievable computational intensity, making them more immune.

The amount of data available as input to these methods will continue to increase. The results presented here show this code is well-suited to handling the requisite I/O, and we expect even more so with the increasing prevalence of burst buffer hardware. However, data management for the large quantities of generated data will require careful workflow design.

This work relies on aggressive asynchronous overlap of communications, transfers and computations in conjunction with double buffering methods for the GPUs. Many applications that have not yet embraced asynchronous approaches will see substantial performance benefits from doing so.

We presently do not require asynchronous task-based methods with dynamic scheduling. Near-peak performance is already achieved, hence benefits from better resource utilization from dynamic task scheduling would be limited. Further, the benefits for resilience would also be limited. Since this is not a time-dependent simulation, steps do not require results from previous steps, thus expensive checkpoint storage is not required to recover from failures, and failed steps can be easily re-run at low cost. Future HPC systems may manifest more severe performance irregularities or failure rates. In this case, the modified GEMMs could be broken into pieces, wrapped in tasks and submitted to a NUMA-aware adaptive DAG scheduler to make better use of hardware resources and address resilience concerns, though this may reduce BLAS-3 computational intensity depending on the implementation. Additionally, the CCC methods considered here possess computational invariants which, with some added computations, could be employed to detect silent errors if needed.

This work has affinities with recent efforts in distributed dense linear algebra, for example, Cyclops [44] and ExaTensor [52] tensor libraries and SLATE [53], all concerned with high performance at scale for distributed dense methods. Vector similarity methods are also used in diverse science domains such as chemistry, image processing, linguistics, ecology and document processing. The methods discussed here are thus potentially applicable beyond the field of genomics.

Acknowledgements

The authors would like to thank Stephen Abbott, Mark Berrill, Matt Ezell, Judy Hill, Annabel Large, Jeff Larkin, Don Maxwell, Verónica Vergara, Sean Treichler, Jack Wells and Chris Zimmer for their assistance. This research used resources of the Oak Ridge Leadership Computing Facility and was supported by the Center for Bioenergy Innovation and the Plant-Microbe Interface SFA (both supported by the Office of Biological and Environmental Research in the DOE Office of Science) at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC, for the US Department of Energy under contract DE-AC05-00OR22725.

REFERENCES

[1] C. S. Florence, C. Zhou, F. Luo, and L. Xu, "The economic burden of prescription opioid overdose, abuse, and dependence in the united states, 2013," *Medical care*, vol. 54, no. 10, pp. 901–906, 2016.

[2] N. E. Morone and D. K. Weiner, "Pain as the fifth vital sign: exposing the vital need for pain education," *Clinical therapeutics*, vol. 35, no. 11, pp. 1728–1732, 2013.

[3] A. Van Zee, "The promotion and marketing of oxycontin: commercial triumph, public health tragedy," *American journal of public health*, vol. 99, no. 2, pp. 221–227, 2009.

[4] C. for Behavioral Health Statistics and Quality, "2015 national survey on drug use and health: Detailed tables," 2016.

[5] "Cdc/nchs national vital statistics system, mortality," https://wonder.cdc.gov, 2017.

[6] K. E. Vowles, M. L. McEntee, P. S. Julnes, T. Frohe, J. P. Ney, and D. N. van der Goes, "Rates of opioid misuse, abuse, and addiction in chronic pain: a systematic review and data synthesis," *Pain*, vol. 156, no. 4, pp. 569–576, 2015.

[7] P. Muhuri, J. Gfroerer, and M. Davies, "Associations of nonmedical pain reliever use and initiation of heroin use in the united states. samhsa: Cbhsq data review," 2013.

[8] T. J. Cicero, M. S. Ellis, H. L. Surratt, and S. P. Kurtz, "The changing face of heroin use in the united states: a retrospective analysis of the past 50 years," *JAMA psychiatry*, vol. 71, no. 7, pp. 821–826, 2014.

[9] R. G. Carlson, R. W. Nahhas, S. S. Martins, and R. Daniulaityte, "Predictors of transition to heroin use among initially non-opioid dependent illicit pharmaceutical opioid users: A natural history study," *Drug & Alcohol Dependence*, vol. 160, pp. 127–134, 2016.

[10] A. M. Vivolo-Kantor, P. Seth, R. M. Gladden, C. L. Mattson, G. T. Baldwin, A. Kite-Powell, and M. A. Coletta, "Vital signs: trends in emergency department visits for suspected opioid overdosesunited states, july 2016–september 2017," *Morbidity and Mortality Weekly Report*, vol. 67, no. 9, p. 279, 2018.

[11] D. F. Weisberg, K. S. Gordon, D. T. Barry, W. C. Becker, S. Crystal, E. J. Edelman, J. Gaither, A. J. Gordon, J. Goulet, R. D. Kerns *et al.*, "Long-term prescription opioids and/or benzodiazepines and mortality among hiv-infected and uninfected patients," *Journal of acquired immune deficiency syndromes (1999)*, vol. 69, no. 2, p. 223, 2015.

[12] S. Szymczak, J. M. Biernacka, H. J. Cordell, O. González-Recio, I. R. König, H. Zhang, and Y. V. Sun, "Machine learning in genome-wide association studies," *Genetic epidemiology*, vol. 33, no. S1, 2009.

[13] A. A. Brown, A. Buil, A. Viñuela, T. Lappalainen, H.-F. Zheng, J. B. Richards, K. S. Small, T. D. Spector, E. T. Dermitzakis, and R. Durbin, "Genetic interactions affecting human gene expression identified by variance association mapping," *Elife*, vol. 3, 2014.

[14] T. A. Manolio, F. S. Collins, N. J. Cox, D. B. Goldstein, L. A. Hindorff, D. J. Hunter, M. I. McCarthy, E. M. Ramos, L. R. Cardon, A. Chakravarti *et al.*, "Finding the missing heritability of complex diseases," *Nature*, vol. 461, no. 7265, p. 747, 2009.

[15] D. G. Clayton, "Prediction and interaction in complex disease genetics: experience in type 1 diabetes," *PLoS genetics*, vol. 5, no. 7, p. e1000540, 2009.

[16] H. Schunkert, I. R. König, S. Kathiresan, M. P. Reilly, T. L. Assimes, H. Holm, M. Preuss, A. F. Stewart, M. Barbalic, C. Gieger *et al.*, "Large-scale association analysis identifies 13 new susceptibility loci for coronary artery disease," *Nature genetics*, vol. 43, no. 4, p. 333, 2011.

[17] M. I. McCarthy, G. R. Abecasis, L. R. Cardon, D. B. Goldstein, J. Little, J. P. Ioannidis, and J. N. Hirschhorn, "Genome-wide association studies for complex traits: consensus, uncertainty and challenges," *Nature reviews genetics*, vol. 9, no. 5, p. 356, 2008.

[18] T. F. Mackay, "Epistasis and quantitative traits: using model organisms to study gene–gene interactions," *Nature Reviews Genetics*, vol. 15, no. 1, p. 22, 2014.

[19] D. L. Aylor, W. Valdar, W. Foulds-Mathes, R. J. Buus, R. A. Verdugo, R. S. Baric, M. T. Ferris, J. A. Frelinger, M. Heise, M. B. Frieman *et al.*, "Genetic analysis of complex traits in the emerging collaborative cross," *Genome research*, vol. 21, no. 8, pp. 1213–1222, 2011.

[20] J. Gonzalez-Dominguez, B. Schmidt, J. C. Kassens, and L. Wienbrandt, "Hybrid cpu/gpu acceleration of detection of 2-snp epistatic interactions in gwas," in *European Conference on Parallel Processing*. Springer, 2014, pp. 680–691.

[21] S. Climer, A. R. Templeton, and W. Zhang, "Allele-specific network reveals combinatorial interaction that transcends small effects in psoriasis gwas," *PLoS Comput Biol*, vol. 10, no. 9, p. e1003766, 2014.

[22] S. Basu, K. Kumbier, J. B. Brown, and B. Yu, "iterative random forests to discover predictive and stable high-order interactions," *Proceedings of the National Academy of Sciences*, p. 201711236, 2018.

[23] S. Mukherjee, S. Kim, V. K. Ramanan, L. E. Gibbons, K. Nho, M. M. Glymour, N. Ertekin-Taner, T. J. Montine, A. J. Saykin, P. K. Crane *et al.*, "Gene-based gwas and biological pathway analysis of the resilience of executive functioning," *Brain imaging and behavior*, vol. 8, no. 1, pp. 110–118, 2014.

[24] S. Climer, W. Yang, L. de las Fuentes, V. G. Da vila Roma n, and C. C. Gu, "A Custom Correlation Coefficient (CCC) Approach for Fast Identification of Multi-SNP Association Patterns in Genome-Wide SNPs Data," *Genetic Epidemiology*, vol. 38, no. 7, 2014, http://www.ncbi.nlm.nih.gov/pubmed/25168954.

[25] D. A. Weighill and D. A. Jacobson, "3-way networks: application of hypergraphs for modelling increased complexity in comparative genomics," *PLoS computational biology*, vol. 11, no. 3, p. e1004079, 2015.

[26] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Advances in neural information processing systems*, 2007, pp. 1601–1608.

[27] W. Joubert, J. Nance, D. Weighill, and D. Jacobson, "Parallel Accelerated Vector Similarity Calculations for Genomics Applications," *Parallel Computing, vol. 75, July 2018, pp. 130-145*, https://www.sciencedirect.com/science/article/pii/S016781911830084X.

[28] D. Jacobson and G. Emerton, "Gsa-pca: gene set generation by principal component analysis of the laplacian matrix of a metabolic network," *BMC bioinformatics*, vol. 13, no. 1, p. 197, 2012.

[29] S. Van Dongen, "Graph clustering via a discrete uncoupling process," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 121–141, 2008.

[30] W. Joubert, J. Nance, S. Climer, D. Weighill, and D. Jacobson, "Parallel Accelerated Custom Correlation Coefficient Calculations for Genomics Applications," *Parallel Computing, to appear*, https://arxiv.org/abs/1705.08213.

[31] W.-H. Wei, G. Hemani, and C. S. Haley, "Detecting epistasis in human complex traits," *Nature Reviews Genetics*, 2014, http://www.nature.com/nrg/journal/v15/n11/full/nrg3747.html.

[32] L. S. Yung, C. Yang, X. Wan, and W. Yu, "GBOOST: a GPU - based tool for detecting gene gene interactions in genome wide case control studies," *Bioinformatics*, vol. 27, no. 9, pp. 1309–1310, 2011. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/27/9/1309.abstract

[33] Q. Wang, F. Shi, A. Kowalczyk, R. M. Campbell, B. Goudey, D. Rawlinson, A. Harwood, H. Ferra, and A. Kowalczyk, "GWISFI: A universal GPU interface for exhaustive search of pairwise interactions in case-control GWAS in minutes," in *2014 IEEE International Conference on Bioinformatics and Biomedicine*, 2014, http://ieeexplore.ieee.org/document/6999192.

[34] J. Gonzalez-Dominguez, S. Ramos, J. Tourino, and B. Schmidt, "Parallel Pairwise Epistasis Detection on Heterogeneous Computing Architectures," *IEEE Transactions on Parallel and Distributed Systems*, 2016, http://ieeexplore.ieee.org/document/7165657/.

[35] J. Gonzalez-Dominguez and B. Schmidt, "GPU-accelerated exhaustive search for third-order epistatic interactions in casecontrol studies," *Journal of Computational Science*, vol. 8, pp. 93 – 100, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877750315000393

[36] B. Goudey, M. Abedini, J. L. Hopper, M. Inouye, E. Makalic, D. F. Schmidt, J. Wagner, Z. Zhou, J. Zobel, and M. Reumann, "High performance computing enabling exhaustive analysis of higher order single nucleotide polymorphism interaction in Genome Wide Association Studies," in *Proceedings of the HISA BIG DATA 2013 Conference*, 2015, https://link.springer.com/article/10.1186/2047-2501-3-S1-S3.

[37] G. R. Luecke, N. T. Weeks, B. M. Groth, M. Kraeva, L. Ma, L. M. Kramer, J. E. Koltes, and J. M. Reecy, "Fast Epistasis Detection in Large-Scale GWAS for Intel Xeon Phi Clusters," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, 2011, http://ieeexplore.ieee.org/document/7345653/.

[38] N. T. Weeks, G. R. Luecke, B. M. Groth, M. Kraeva, L. Ma, L. M. Kramer, J. E. Koltes, and J. M. Reecy, "High-performance epistasis detection in quantitative trait GWAS," *The International Journal of High Performance Computing Applications*,

vol. 0, no. 0, p. 1094342016658110, 0. [Online]. Available: http://dx.doi.org/10.1177/1094342016658110

[39] J. Gonzalez-Dominguez, J. C. Kassens, L. Wienbrandt, and B. Schmidt, "Large-scale Genome-wide Association Studies on a GPU Cluster Using a CUDA-accelerated PGAS Programming Model," *Int. J. High Perform. Comput. Appl.*, vol. 29, no. 4, pp. 506–510, Nov. 2015. [Online]. Available: http://dx.doi.org/10.1177/1094342015585846

[40] J. C. Kassens, J. Gonzalez-Dominguez, L. Wienbrandt, and B. Schmidt, "UPC++ for bioinformatics: A case study using genome-wide association studies," in *2014 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2014, pp. 248–256.

[41] I. S. Haque, V. S. Pande, and W. P. Walters, "Anatomy of High-Performance 2D Similarity Calculations," *Journal of Chemical Information and Modeling*, vol. 51, no. 9, pp. 2345–2351, 2011. [Online]. Available: http://dx.doi.org/10.1021/ci200235e

[42] S. Tomov, R. Nath, H. Ltaief, and J. Dongarra, "Dense linear algebra solvers for multicore with GPU accelerators," in *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, April 2010, pp. 1–8, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5470941.

[43] J. Choi, J. J. Dongarra, R. Pozo, and D. W. Walker, "ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers," in *Frontiers of Massively Parallel Computation, 1992., Fourth Symposium on the*. IEEE, 1992, pp. 120–127, http://ieeexplore.ieee.org/document/234898/.

[44] E. Solomonik, D. Matthews, J. Hammond, and J. Demmel, "Cyclops Tensor Framework: Reducing communication and eliminating load imbalance in massively parallel contractions," in *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, 2013, pp. 813–824, http://ieeexplore.ieee.org/document/6569864/?arnumber=6569864.

[45] V. Anatharaj, F. Foertter, W. Joubert, and J. Wells, "Approaching Exascale: Application Requirements for OLCF Leadership Computing," *Oak Ridge National Laboratory, Technical Report ORNL/TM-2013/186*, 2013, https://www.olcf.ornl.gov/wp-content/uploads/2013/01/OLCF_Requirements_TM_2013_Final1.pdf.

[46] J. Dongarra, "Experiments with Energy Savings and Short Precision," in *Salishan2018 Proceedings*.

[47] W. Joubert, R. K. Archibald, M. A. Berrill, W. M. Brown, M. Eisenbach, R. Grout, J. Larkin, J. Levesque, B. Messer, M. R. Norman, and et al., "Accelerated application development: The ORNL Titan experience," *Computers and Electrical Engineering*, vol. 46, May 2015.

[48] "The Extreme Heterogeneity Virtual Workshop 2018," https://www.orau.gov/ExHeterogeneity2018/agenda.htm.

[49] "PLASMA," http://icl.cs.utk.edu/plasma/software.

[50] "BLIS," https://github.com/flame/blis.

[51] "OpenBLAS: An optimized BLAS library," http://www.openblas.net/.

[52] D. I. Lyakh, "Portable Heterogeneous High-Performance Computing via Domain-Specific Virtualization," *4th ADAC Workshop*, 2017, https://iadac.github.io/events/adac4/liakh.pdf.

[53] Innovative Computing Laboratory, "SLATE," http://icl.utk.edu/slate/.

Appendix I: Tensor Core Implementation

As described in [30], the CCC method can be implemented by representing the input data as vectors containing entries of two bits and operating on these entries using binary operations followed by a tally of the results for each vector pair (or triple) into a small table. This is illustrated in Figures 7 and 8. For all 4 (or 8) paths through the 2 (or 3) 2-bit values, the occurrences of the associated bit patterns are tallied into the associated table. After the tally, the results are normalized by single-vector bit counts and presented as a set of 4 (or 8) floating point values for each vector pair (triple).

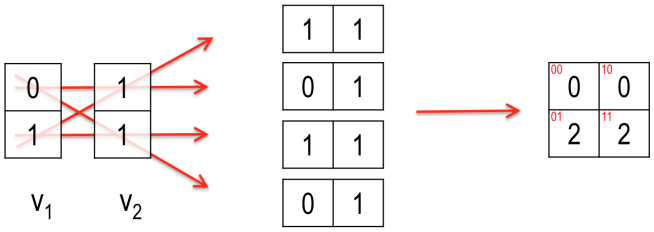

Fig. 7: 2-way CCC calculation example. Left: two vectors of length 1, each entry 2 bits. Center: enumeration of all pairings of entries. Right: tallying of counts of each pairing type (see [30])

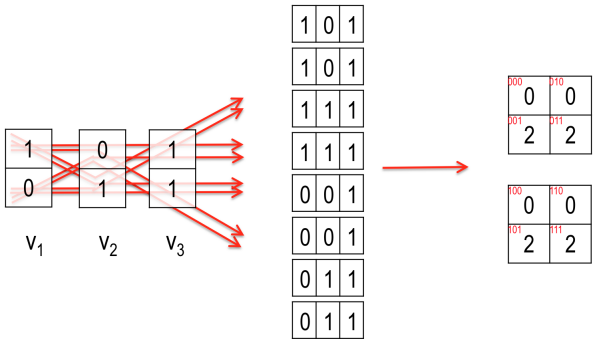

Fig. 8: 3-way CCC calculation example. Left: three vectors of length 1, each entry 2 bits. Center: enumeration of all combinations of entries. Right: tallying of counts of each combination type (see [30])

This computation can, however, alternatively be represented as a standard matrix-matrix product operation. To illustrate we use the example shown in Figure 7. The left and right matrices for this example each with one vector of one element are represented in binary as $A = [[01_2]]$¡ $B = [[11_2]]$. To adapt to the Tensor Core method, new matrices are then formed by doubling the number of columns to represent the number of 0 bits and 1 bits in each entry: $\bar{A} = [[1][1]]$, $\bar{B} = [[0][2]]$. The result is then obtained by forming the product $\bar{B}^T\bar{A}$. In the general case, the result is a matrix of 2×2 blocks with each block representing the comparison of two of the original vectors. See Figure 9.

The full method proceeds as follows:

1) A CUDA kernel copies the original matrix into a new matrix on the GPU composed of FP16 entries, each entry either 0, 1 or 2, as illustrated in Figure 9. For the sparse case and for the case of the 2-way operation that is used to support the 3-way method, the counts are adjusted accordingly. This computation is $O(mn)$.
2) A call to `cublasGemmEx` is made to calculate the product. This takes FP16 inputs, calculates in FP32 arithmetic and accumulates the results in FP32. FP32 is required rather than FP16 in order to maintain sufficient accuracy. The calculation is exact for vectors up to length $m = 2^{22} - 1$, roughly four million, and with roundoff maintains sufficient accuracy for sizes larger than this. The computation is $O(mn^2)$.
3) The result of the product is adjusted in-place to the data format required by the host code, using a CUDA kernel running in $O(n^2)$ time.

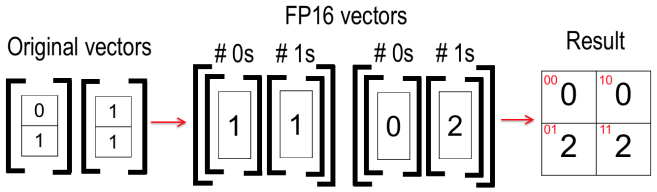

Fig. 9: 2-way Tensor Core CCC calculation example. Left: two vectors of length 1, each entry 2 bits. Center: doubling of columns to represent per-element bit counts. Right: matrix-matrix product.

Implementation measures are required to maximize performance, including: ensuring that matrix dimensions are divisible by a small power of two by use of padding; setting the matrix axis order (row major vs. column major) to increase locality; and requesting the best algorithm selection for the `cublasGemmEx` call based on results of empirical testing.

Since the copy of A, B to $\bar{A}$, $\bar{B}$ requires a memory expansion of 16X, a code option is provided to divide the process described above into steps, based on decomposing A and B into block rows and operating on a pair of block rows at a time, accumulating the result. This reduces memory pressure from converting the matrices to FP16 all at once.

The copying to new matrices increases the number of entries of the original matrix pencil $[A, B]$ by a factor of two and thus the number of effective pairwise vector comparisons by a factor of four. However, because of the speed the Tensor Cores (16X faster than double precision performance for the same operations), the overall impact is a significant speedup compared to the original bitwise method.

Appendix II: System Execution Configuration

Titan is a Cray XK7 system composed of 18,688 compute nodes, each equipped with an AMD Interlagos 16 core CPU and an Nvidia Kepler K20X GPU connected via a PCIe-2 bus. The K20X GPU has peak single (double) precision flop rate of 3.935 (1.311) TF and peak memory bandwidth of 250 GB/sec. Each node contains 32 GB main memory and 6 GB GDDR GPU memory. Titan is equipped with a Gemini 3D torus interconnect. The Lustre file system Atlas is used for I/O. The software versions used are Cray OS version 5.2.82, Cray Programming Environment 2.5.13, GCC 4.9.3, MAGMA

1.6.2 and CUDA toolkit 7.5.18-1.0502.10743.2.1. The environment variable `APRUN_BALANCED_INJECTION` and additionally the variables `ARMCI_DMAPP_LOCK_ON_GET` and `ARMCI_DMAPP_LOCK_ON_PUT` are used in some cases to improve communication throughput, as well as `MPICH_RANK_REORDER_METHOD` with a random ordering.

Summit is composed of 4,608 compute nodes, each with two 22-core IBM POWER9 processors and six Nvidia Volta V100 GPUs each connected to a POWER9 by NVLINK-2 with peak performance 100 GB/sec bidirectional. V100 GPU peak single (double) precision performance is approximately 14 (7) TF and peak memory bandwidth 900 GB/sec. Each node contains 512 GB main memory, while each GPU contains 16 GB HBM2 memory. The nodes are connected with a Mellanox Infiniband fat tree interconnect. Each node is equipped with a 1.6 TB NVMe burst buffer device. As of this writing, Summit is connected to the GPFS file system AlpineTDS while the final Alpine file system is being prepared for production use. Software versions used are GCC 6.4.0, MAGMA 1.6.2, Spectrum MPI 10.2.0.0 and CUDA 9.2.88. The `jsrun` tool is used for application launch, with six MPI ranks per node each with 1 GPU and 7 OpenMP threads mapped to CPU cores. Environment variables `PAMI_IBV_ENABLE_DCT=1` `PAMI_ENABLE_STRIPING=1` `PAMI_IBV_ADAPTER_AFFINITY=0` `PAMI_IBV_QP_SERVICE_LEVEL=8` `PAMI_IBV_ENABLE_OOO_AR=1` are set to use adaptive routing and to limit per-node storage needed for MPI buffers.

As of this writing, Summit has not fully completed acceptance testing; therefore, all results are preliminary, and performance may improve in the future as the system undergoes more performance tuning.