Software project ’miraculix’: 
Efficient computations with large genomic datasets

Martin Schlather

Universität Mannheim
Institute for Mathematics

joint work with
Alexander Freudenberg, Guido Moerkotte,
Torsten Pook and Jeremie Vandenplas

Funding: Analyses for this work were performed on the HPC system bwUniCluster funded by the state of Baden-Württemberg

Lyon 2023
Idea and motivation behind ‘miraculix’

- miraculix is a library, not a standalone package
- goals
  - improving time (and/or memory) critical parts of code used in genetics
  - detection of hardware during run-time
- close cooperations with partners at Wageningen and Göttingen
- code published irregularly on github

CPU solutions are represented here
  GPU solutions will mainly be presented in the next talk
Integration of miraculix

MoBPS uses miraculix for

- Compressed storage of genomic datasets (twobit format)
- Efficient calculation of the genomic relationship matrix
- Computation of BVE for gBLUP models (Cholesky decomposition)

→ Breeding program simulation is significantly accelerated (Pook et al., 2021)

MiXBLUP uses miraculix for

- Acceleration of genotype matrix multiplications
- Faster iteration times in solving single-step models

→ BVE in single-step models is substantially faster (Freudenberg et al., 2023b)
Mixed Model Equations

- Goal: Solving single-step Mixed Model Equations, e.g., ssGBLUP:
  \[
  \begin{pmatrix}
  X^T R^{-1} X & X^T R^{-1} W \\
  W^T R^{-1} X & W^T R^{-1} W + H^{-1}
  \end{pmatrix}
  \begin{pmatrix}
  \hat{b} \\
  \hat{u}
  \end{pmatrix} =
  \begin{pmatrix}
  X^T R^{-1} y \\
  W^T R^{-1} y
  \end{pmatrix}.
  \]

- Solver software uses iterative algorithms (e.g., PCG)
- Each iteration requires multiplication of the coefficient matrix
- Coefficient matrix involves the genotype matrix \( Z \) and its transposed \( Z^T \)
- for ease, \( Z[...]Z^T V \), but also \( ZZ^T \) will be considered in the following
Time Results $ZZ^\top$

Time measurements for GRM calculation with 10k individuals

Hardware: Xeon Platinum 8368

Time for reading and writing is not measured when miraculix is used.

→ An additive constant in the computing times of miraculix is missing.
Computing times for $Z(Z^\top V)$

Left:
- 50k individuals
- small/medium/large = 102k/700k/3100k snps
- AMD Milan EPYC 7513 (20 cores)

Right:
- 50k individuals
- small/medium/large = 102k/700k/3100k snps
- XEON 6230
Frame conditions

- Libraries for double-double scalar products are well-developed
  - Current approaches decompress packed 2-bit representation (of plink)
- Transformation of data is always worth, as soon as computing time is of smaller order.
- Additional memory are considered acceptable (once or twice the original size)
Hash table calculating $Z Z^\top$ (based on plink coding)

**Basis:** SIMD command `mm_shuffle_epi8`

- hash table: 16 entries, 1 byte each
- addressed simultaneously by lower $\frac{1}{2}$ byte of each byte in the register
- 16 hash table look-ups at once for SIMD (64 for AVX512)

**Goal:** find operator $\circ$ with identifiable results

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$00_p$</td>
<td>$10_p$</td>
<td>$11_p$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$00_p$</th>
<th>$10_p$</th>
<th>$11_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

operator $\circ$ is a composition of bitwise $\&$, $|$, $\gg$, and a substraction
Idea for calculating $Z(Z^\top V)$ for AVX2 only

**Basis: Hash table in the L1 cache**
- hash table: 243 entries, 1 double each

**Mathematical background**

$Z_1, \ldots, Z_5 \in \{0, 1, 2\}$: arbitrary SNP values
$V_1, \ldots, V_5 \in \mathbb{R}$, fixed

Scalar product of $(Z_1, \ldots, Z_5)$ with $(V_1, \ldots, V_5)$,

$$Z_1 V_1 + \ldots + Z_5 V_5,$$

takes only $3^5 = 243$ different values.

**Idea**
- compress 5 SNP values into 1 byte
- this byte addresses the result of the scalar product in the hash table
References


