Research focus: low-rank compression in mixed precision for the solution of sparse linear systems

**LOW-PRECISION ARITHMETICS**

<table>
<thead>
<tr>
<th>Mantissa (m)</th>
<th>Exponent</th>
<th>Range</th>
<th>μ = 2^{-m-1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>fp64 (double)</td>
<td>52 bits</td>
<td>11 bits</td>
<td>10^{-308} - 1 x 10^{-16}</td>
</tr>
<tr>
<td>fp32 (single)</td>
<td>23 bits</td>
<td>8 bits</td>
<td>10^{-126} - 6 x 10^{-8}</td>
</tr>
<tr>
<td>fp16 (half)</td>
<td>10 bits</td>
<td>5 bits</td>
<td>10^{-32} - 5 x 10^{-4}</td>
</tr>
<tr>
<td>bfloat16 (half)</td>
<td>7 bits</td>
<td>8 bits</td>
<td>10^{-38} - 4 x 10^{-3}</td>
</tr>
</tbody>
</table>

**BLR MATRICES**

We consider a certain class of matrices, whose off-diagonal blocks have low numerical ranks. More precisely, the singular values of such blocks decrease rapidly, typically following an exponential decay. BLR compression consists in approximating each of those blocks as a product of two smaller rectangular matrices (low-rank approximation). It may be based on a truncated SVD or QR decomposition.

![Example of a BLR matrix](image)

Low rank approximation of a block: B ≈ X × Y

Color scale: numerical ranks of the blocks.

**LOW-RANK APPROXIMATION IN MIXED PRECISION**

We introduce a new approach to handle a low-rank approximation, in case it is based on a truncated SVD or QR decomposition. We propose to separate the columns into several groups, associated with different floating-point formats.

- A criterion for storing columns xₖ and yₖ in precision fp₃₂:
  \[ \frac{||B||}{\sigma_{max}} < \epsilon \leq \frac{1}{\sqrt{\epsilon_{fp32}}} ||B|| \]
- Compression error: \[ ||B - X\Sigma Y|| \leq 5\epsilon ||B|| \], instead of \[ \epsilon ||B|| \]

\[ X_1X_2X_3 \Sigma_1\Sigma_2\Sigma_3 \]

Approximation of a block as a truncated SVD

![Repartition of the singular values of a block: a typical example](image)

**LU FACTORIZATION (DENSE MATRICES)**

- Block LU factorization algorithm, step k:
  \[ \rightarrow \text{Compute } L_kU_k = A_{ik} \]
  \[ \rightarrow \text{Update formula: for } i,j > k, A_{ij} \leftarrow A_{ij} - \left( A_{ik}U_{kj}^{-1} \right) \times \left( L_{kj}^{-1}A_{kj} \right) \]
- With BLR compression, the approximation \[ A_{ik} \approx X_{ik}Y_{ik}^T \] allows to reduce the number of operations.
- Example of kernel in mixed precision: multiplication LR × matrix:

![Representation of a low-rank block stored in 7 precisions](image)

- Our first results show that, by adding mixed precision, there is a gain of storage between 1.12 and 1.17 regarding the LU factors:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>precision</th>
<th>Factor size (GBytes)</th>
<th>Memory peak (GBytes)</th>
<th>Scaled residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>thingas</td>
<td>fp64</td>
<td>95</td>
<td>120</td>
<td>6.4E-14</td>
</tr>
<tr>
<td></td>
<td>mixed</td>
<td>59</td>
<td>86</td>
<td>5.5E-14</td>
</tr>
<tr>
<td>perf009</td>
<td>fp64</td>
<td>25.6</td>
<td>36</td>
<td>1.3E-10</td>
</tr>
<tr>
<td></td>
<td>mixed</td>
<td>20.5</td>
<td>32</td>
<td>1.4E-10</td>
</tr>
</tbody>
</table>

**RESULTS ON DENSE MATRICES**

- We emulate a LU factorization with BLR in 3 precisions: fp64, fp32 and bfloat16.
- Hypothesis: time cost = flops(fp64) + \( \frac{1}{2} \)flops(fp32) + \( \frac{1}{4} \)flops(bfloat16)
- We plot the relative gains with mixed precision compared to double precision, as a function of the error. We notice that, for a given error, the mixed precision variant achieves better performances than the double precision (\( \times 2 \) to \( \times 4 \) in terms of storage and expected time).

![Comparison of LU factorization time](image)

**LU FACTORIZATION OF SPARSE MATRICES**

A multifrontal solver, such as MUMPS[3], computes a LU factorization of a sparse matrix. In order to do that, many partial LU factorizations of smaller dense matrices are computed.

The LU factors are potentially BLR matrices, and their storage cost is a large part of the memory peak. We added an option in MUMPS that converts the low-rank blocks to mixed precision when they are not used.

If mixed-precision BLR is used for storage gains only, only a conversion operation is needed for the formats. Instead of the 3 common floating-point formats, we decided to use 7, respectively on 64, 56, 48, 40, 32, 24, and 16 bits.

- Example: conversion from fp32 to "fp24":

**PERSPECTIVES**

- Aim for times gains in MUMPS by performing computations in mixed precision
- Develop a variant of the algorithm that uses fp16 instead of bfloat16. Scaling methods will be required.
- A QR factorization algorithm may be accelerated using mixed precision

**REFERENCES**