Toward hardware support for Reproducible BLAS
http://bebop.cs.berkeley.edu/reproblas/

James Demmel, Hong Diep Nguyen

SCAN 2014 - Wurzburg, Germany

Sep 24, 2014
Reproducibility

Reproducibility: obtaining bit-wise identical results from different runs of the program on the same input data, regardless of different available resources.

Cause of nonreproducibility: not by roundoff error but by the non-determinism of accumulative roundoff error.

Due to the non-associativity of floating point addition, accumulative roundoff errors depend on the order of evaluation, and therefore depend on available computing resources.
Reproducible Summation

\[ s = \sum_{1}^{N} v[i] \]

Error bound: \[ |s - \sum_{1}^{N} v[i]| < N \times \epsilon \times \sum_{1}^{N} |v[i]|. \]

Running error depends on the order of evaluation.

Solutions:

- Increasing the accuracy (Kahan’s algorithm, distillation algorithm, extra precision, \ldots) can increase the chance of reproducibility but does not guarantee reproducibility.
- Exact arithmetic or correctly-rounded algorithm can provide reproducibility: costly both in terms of memory and computation
- Our proposed solution: pre-rounding technique
Reproducible Summation: Pre-rounding technique

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad \ldots \]

\[ \text{proc 1} \quad \text{proc 2} \quad \text{proc 3} \]

---

Reproducible Summation: Pre-rounding technique

\[ \text{EMAX} \quad \text{W-bit} \quad \text{EMIN} \]

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad \ldots \]

\[ k \times W \]

1

\[ ^1\text{S.M. Rump, Ultimately Fast Accurate Summation, SIAM Journal on Scientific Computing (SISC), 2009} \]
Indexed Floating-Point Format

**Idea**: representing the partial sum by:

- the index of the left-most bin:

  \[ \text{width(Index)} \geq \log_2 \left( \frac{\text{EMAX} - \text{EMIN}}{W} \right) \]

- \( K \) numbers of \( BW \) bits to represent the \( K \) left-most bin. Maximum number of addends that can be added without overflow:

  \[ N_{\text{max}} = 2^{BW-W-1-1} \]

- Absolute error bound:

  \[ |S - \sum_{i=1}^{N} v_i| \leq N \times \text{ulp(last bin)} = N \times 2^{-(K-1)W} \times \max_{1}^{N} |v_i| \]
ReproBLAS \url{http://bebop.cs.berkeley.edu/reproblas}

ReproBLAS is a library for (Parallel and Sequential) Reproducible Basic Linear Algebra Subroutines, currently only supports level-1 routines for 4 basic data type (single/double precision, real/complex numbers)

Configuration for double precision: $W = 40, K = 3$

- Can accumulate up to $2^{(P-1)-W-1} = 2^{11} = 2048$ numbers in mantissa part without any overflow.
- Can accumulate $2^{2*P-W-2} = 2^{64}$ numbers using carry part.
- The absolute error bound in the worst case is $N \times 2^{(K-1)*W} \times \max |v_i| = 2^{-80} \times N \times \max |v_i|$

- **Require only one reduction operation.**
- Run $8 \times$ slower than performance-optimized library on a single processor, but only $1.2 \times$ slower on massively parallel environment such as CRAY XC30 machine with 1024 processors.
Hardware support

**Goals:**
- Reduce the slowdown of reproducible operations on single processor to as close to $1 \times$ as possible,
- Require minimal changes to current hardware,

**Approaches:**
- Dedicated Accumulator
- New instructions to support the implementation of reproducible addition:
  - Using existing 128-bit/256-bit register to represent indexed floating-point format,
  - Using existing load-store instructions,
  - Can be pipelined, multi-threaded.
Instructions

- Addition
  - a native floating-point to an indexed floating-point number
  - two indexed floating-point numbers

- Conversion
  - From native floating-point number to indexed numbers: implicitly through the addition
  - From indexed format to native format: not frequently used, can be implemented in software

- Carry-bit propagation: propagate the overflow bit to a higher order register to increase the maximum number of addends.
Data Format Layout

Requirements:

- $\text{width(Index)} + K \times BW \leq \text{register width}$
- $BW > W$
- $\text{width(Index)} \geq \log_2 \left( \frac{E_{\text{MAX}} - E_{\text{MIN}}}{W} \right)$
- Reasonable error bound:

$$|S - \sum_{1}^{N} v_i| \leq N \times 2^{-(K-1)W}$$

For double precision floating-point number, using 128-bit register:

- $\text{width(Index)} + K \times BW \leq 128$
- $\text{width(Index)} \geq 11 - \lceil \log_2(W) \rceil$

Configuration: $K = 3, W = 32, BW = 40, \text{width(Index)} = 8$
128-bit Indexed Floating-Point Format

Configuration: $K = 3$, $W = 2^5$, $BW = 40$, $width(I) = 8$
128-bit Indexed Floating-Point Format

Configuration: $K = 3$, $W = 2^5$, $BW = 40$, $\text{width}(I) = 8$
128-bit Indexed Floating-Point Format

Configuration: $K = 3$, $W = 2^5$, $BW = 40$, $\text{width}(I) = 8$
128-bit Indexed Floating-Point Format

Configuration: \( K = 3, \ W = 2^5, \ BW = 40, \ \text{width}(I) = 8 \)
Properties

Absolute error bound in the worst case:

$$|s - \sum_{1}^{N} v[i]| \leq N \times 2^{(K-1)\times W} \times \max |v[i]| = 2^{-64} \times N \times \max |v[i]|$$

Number of additions that can be performed without overflow:

$$N_{max} = 2^{40-32-1-1} = 64$$
**Objective**: ensure that there will be no overflow over the next $2^{B_W-W-1-1}$ additions. Using the same format for the carry register:

$$N_{max} = 2^{B_W-W-1-1} \times 2^{B_W-2} = 2^{45}$$
Experimental results

Simulation in software:

- Implemented in Chisel, a scala-based programming language for hardware construction.
- Operations:
  - RAdd: add 1 double precision FP to an 128-bit Indexed Floating-Point,
  - RAddR: add 2 128-bit Indexed Floating-Point,
  - RRenorm, RCary: perform bit propagation to avoid overflow.
  - Each operation can be executed in 1 clock cycle
- No exception-handling
- ≈ 230 LOC for the hardware construction
- ≈ 400 LOC for testing and validation
Sequential summation of $N$ double precision floating-point numbers

```c
int i, NB = 64;
Idouble s, c;
for (iN = 0; iN < n; iN += NB) {
    for (i = iN; i < min(n, i+NB); i++) {
        s = RAdd(s, v[i]);
    }
    c = RCarry(c, s);
    s = RRernorm(s);
}

Cost: $N + \mathcal{O}\left(\frac{N}{NB}\right)$ FLOPs
Reproducible Sum: Accuracy

\[ v[i] = \sin(2.0 \times \pi \times i/N), \quad N = 10^5 \]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$1 \rightarrow N$</th>
<th>$N \rightarrow 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>quadruple</td>
<td>9.923413837157274E-15</td>
<td>9.92341383715682E-15</td>
</tr>
<tr>
<td>Reproducible</td>
<td>9.923377224108076E-15</td>
<td>9.923377224108076E-15</td>
</tr>
<tr>
<td>Normal Sum</td>
<td>5.513115788968589E-13</td>
<td>3.0460904930145957E-12</td>
</tr>
</tbody>
</table>
New instructions:

- Operates on existing 128-bit register file,
- Executes in 1 single cycle,
- Requires no change to the scheduling system,
- Helps to reduce the cost of reproducible summation to $\approx N$ FLOPs, and is almost as accurate as the normal summation algorithm.
Implementation on real hardware to collect real data on required area as well as the energy consumption of proposed instructions.

Fused Multiply-Add support

Implementation for hardware without support of 128-bit register.

Implementation of BLAS level 2, 3 routines.

Implementation of software library that provides exactly the same results as those computed using the newly proposed instructions.