ARM Performance Libraries – Current and future interests

ARM

Chris Goodyer
Senior Engineering Manager, HPC Software

Workshop on Batched, Reproducible, and Reduced Precision BLAS
25th February 2017

ARM Performance Libraries

Commercial 64-bit ARMv8 math libraries

- Commonly used low-level math routines - BLAS, LAPACK and FFT
- Validated with NAG's test suite, a de-facto standard

Performance on par
with best-in-class math libraries

Best-in-class performance with commercial support

- Tuned by ARM for Cortex-A72, Cortex-A57 and Cortex-A53
- Maintained and Supported by ARM for a wide range of ARM-based SoCs
- Regular benchmarking against open source alternatives

Commercially Supported
by ARM

Silicon partners can provide tuned micro-kernels for their SoCs

- Partners can collaborate directly working with our source-code and test suite
- Alternatively they can contribute through open source route

Validated with
NAG test suite

ARM's mission

- Deploy energy-efficient ARM-based technology, wherever computing happens…

Leading in wearables and the Internet of Things

~85% share of laptops, tablets, and smartphones

Driving the transformation of the network and data center to an Intelligent Flexible Cloud

Enabling innovation and creativity with embedded intelligence

Taking mobile computing to the next four billion people

Partnering to deliver data center efficiency

ARM

ARM are big supporters of standards

- One of the reasons the ARM ecosystem has been so successful in both mobile and embedded spaces is that we actively support and encourage standards
- The ARM model of working with partners to enable them to be successful means that we want to help everyone
  - That comes from minimizing cost of switching between providers
- Software standards, such as BLAS, are a vital part of this
  - BLAS is successful because everyone has adopted it
  - Vendor specific extensions are risky as it reduces portability of end-user code
- End users are the only people that matter!
  - They (typically) don't get to choose the architecture that is purchased
  - They just have to make sure their codes run as well as possible on them
  - This is why we don't have a sparse BLAS yet
    - We need a standard that all vendors and the major packages will want to use

Batched BLAS – finalized standard needed!

- ARM is an emerging player in HPC and server workloads
  - For now, we have limited end-user deployment which gives us added flexibility to wait
  - We appreciate other vendors have been able to drive the need by providing implementations ahead of the standard
  - Important now to encourage users to use the agreed standard rather than legacy interface
- As such we have not rushed to get a Batched BLAS interface included, rather waiting until the standard is settled
- Wider adoption and usage may require codes, such as TensorFlow, adopting it

Reduced precision – FP16

- Reduced precision will be of great interest to many classes of our customers
  - Computer vision and machine learning are two such areas
- ARMv8.2 optionally supports the IEEE half-precision floating-point format as a first-class data type for all scalar and Advanced SIMD floating-point computational instructions.
  - The optionality is significant: some ARMv8.2 processors may not implement FP16
    - Software will need to be able to fall back to using FP32 arithmetic with FP16 only as a storage format when it isn't available.
- ARM's Scalable Vector Extension (SVE) includes FP16
  - Potential benefit is doubling the throughput of SIMD floating-point, though that does depend on how FP16 SIMD has implemented
    - i.e. can it perform 2nxFP16 multiplies in no more cycles than nxFP32 multiplies?
    - Note: ARMv8-A's Advanced SIMD (aka NEON) has 128-bit vectors, SVE is **up to** 2048 bits

Reduced precision BLAS – questions from a vendor

- The big question for BLAS in half precision is "Does it all need to be done?"
  - This is then followed by "if BLAS, why not LAPACK"
- Compiler support, both commercial and open source, is important
  - At present we are not imagining existing HPC codes will want half precision
- It can all be implemented, but will most of it be used for real work?
  - Will people pay for it?
- Half precision versions of GEMMs will be vital for deep learning
  - The likelihood of wanting half precision is that absolute accuracy is less important

Reproducibility from BLAS

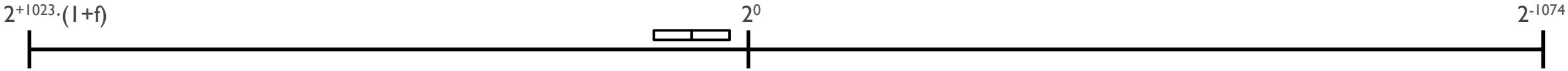
- ARM Performance Libraries are [should be!] reproducible between:
  - consecutive runs on with same input run on same number of identical cores
- If a system was developed with multiple micro-architectures (e.g. ARM big.LITTLE) then different vector lengths will give differences
- We are adopting some OpenMP task-based parallelism into LAPACK (c.f. PLASMA) which may, *by default*, change reproducibility
  - If patches get updated by multiple tasks then ordering of updates will make differences
  - Adding user control for reproducibility is not a current demand from users, but may be in future

Future possible architectural reproducibility feature

- Last year I talked a little about the High Precision Accumulator
- This year a quick refresh and some notes on developments…

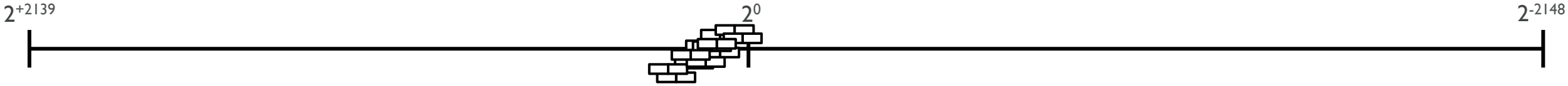
- What our architecture team would like is an indication of interest from the scientific community
- From a BLAS point of view an operation like DDOT could be made effectively associative, hence parallelizable, reproducible and accurate

IEEE FP exponent range



- IEEE fp64: 11 bit exponent: $2^{-1074} \le$ fp64 value $\le 2^{+1023}\cdot(1+f)$
- fp64 numbers have 53-bit significands
- unrounded fp64 products have 106-bit significands

- Kulisch accumulator spans lsb of P_{min} to msb of P_{max} with 92 extra bits' "headroom"



- $P_{min} = (fp64_{min})^2 = 2^{-2148}$; $P_{max} = (fp64_{max})^2 = 2^{+2047}\cdot(1+f)$

High-Precision Anchored (HPA) Numbers

- An HPA number comprises:
  - a long 2's-complement integer, comprising 100-200 or even more bits
  - more precision than available in binary64 (or even binary128 if wanted)
  - an anchor that says how to interpret those digits
- e.g. subatomic values could be in the range 2^-100 to 2^-1 (anchor = -100)
- e.g. astronomical values could be in the range 2^20 to 2^199 (anchor = +20)

the arithmetic works the same in both ranges

- A programmer picks the range for the application area or problem
  - Anchor is analogous to a block floating-point FP exponent, i.e. fixed for a given problem
  - Anchor basically represents the least significant exponent value we are interested in
  - The length of the long integer then gives us a range over which we can accumulate exactly
- Eliminating rounding and overflow allows HPA accumulation to be associative

High-Precision Anchored (HPA) format

- New datatype, denoted (Vi, Vm)
  - vector of 64-bit integers considered as one long redundantly-represented integer
  - metadata (either vector or scalar) interpreting long integer – anchor point(s) & overlap
- Example: 232-bit HPA number, with bit-weightings from 155 to -76
  - long long i[3:0]; // four 64-bit values
  - long long m[3:0] = {92, 36, -20, -76}; // boundaries, with 8-b overlaps
  - the high order bit of i[3] represents 2^{155}
  - the low order bit of i[0] represents 2^{-76}
  - could sum FP values corresponding to that range exactly
- Metadata doesn't have to be a vector
  - e.g. could specify long integer l.s.b. weight and overlap / lane
  - vector metadata would increase performance

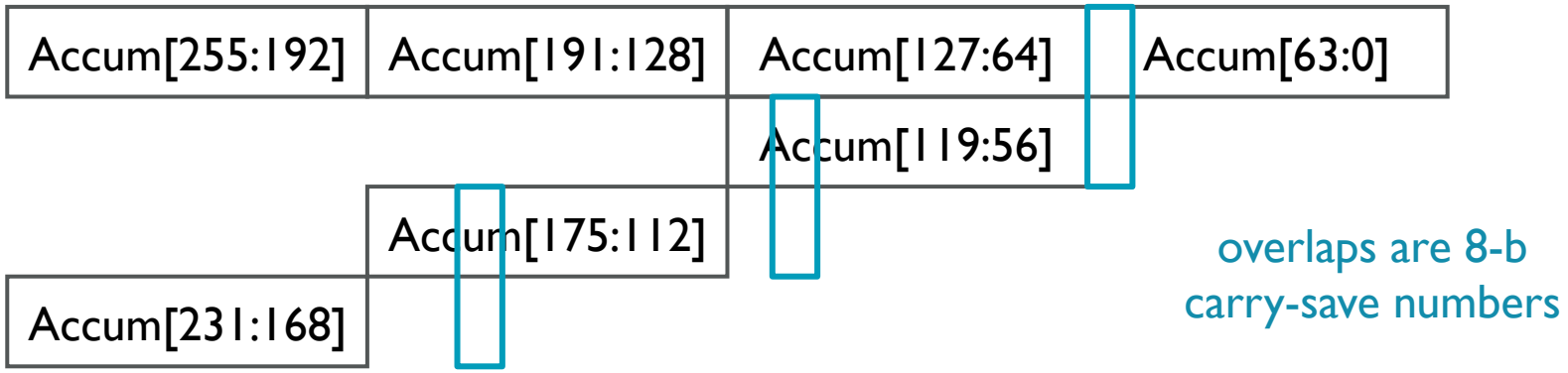
Long Integer Arithmetic

- Central concept: treat vector of 64-bit values as one long integer
- Long integer addition is associative
- Example: 256-bit integer
  - long long a[3:0], b[3:0];

Accum[255:192]	Accum[191:128]	Accum[127:64]	Accum[63:0]

Redundant Long Integer Arithmetic

- Allow vector elements to "overlap"
- For example, allowing 8 bits' overlap between lanes:



- Provide headroom in each lane to accommodate carries
- Treat each lane as a 2's-complement number

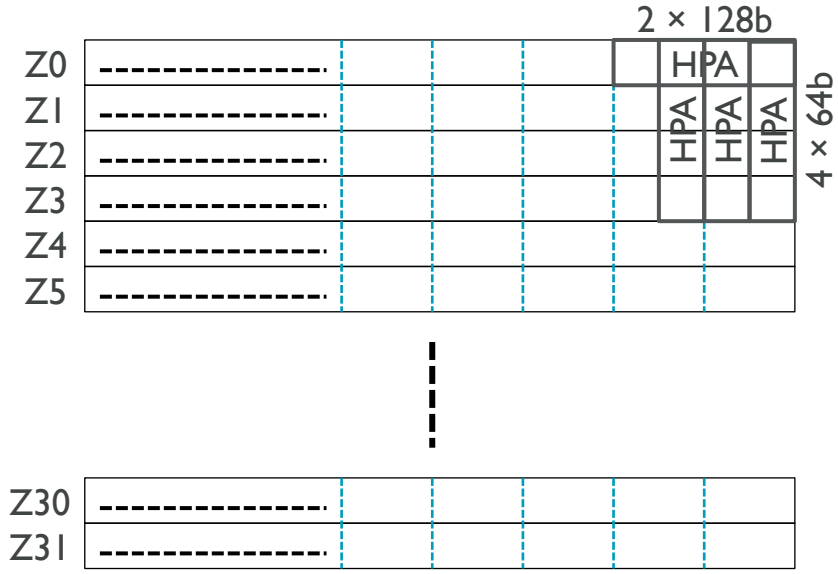
FP Accumulation

- Imagine a hypothetical implementation where converting is implemented in a single cycle
  - You can then add n items in n+1 cycles
- These adds are associative, so no dependencies
  - ∴ fully parallelizable
  - (Suitable for GPUs?)
- Establishing exponent range is the only additional task for a programmer

"Cycle"	1	2	3	4	5
ADD_HPA_FP (Vi,Vm,F1)	Convert	Add			
ADD_HPA_FP (Vi,Vm,F2)		C	A		
ADD_HPA_FP (Vi,Vm,F3)			C	A	
ADD_HPA_FP (Vi,Vm,F4)				C	A

SVE

- 32 scalable vector registers
  - 128·k bits: k = 1...16
- "Vector Length Agnostic"
  - Software knows VL & "self-adjusts" degree of parallel processing accordingly
- Mitigates against HPA?
  - HPA explicitly ***not*** vector length agnostic!
- Solution: place HPA accumulator across several vector registers
  - Process as many HPA accumulators in parallel as scalable vector allows
    - HPA is associative...



Faster FP Accumulation

- These adds are associative, so add in any order, on any number of accumulators...
  - ... on any number of cores
- Here, adding 4n items in n+2 cycles on two accumulators

"cycle"	1	2	3	4
ADD_HPA_FP (Vi,Vm, F1,F5)	C	A		
ADD_HPA_FP (Vj,Vm, F2,F6)	C	A		
ADD_HPA_FP (Vi,Vm, F3,F7)		C	A	
ADD_HPA_FP (Vj,Vm, F4,F8)		C	A	
VADD_HPA_FP (Vj,Vm,Vi)				A

HPA and BLAS

- Example method for utilizing an HPA in a DDOT

```
call blas_ddot_hpa_initialize(hpa_anchor, hpa_len)
call blas_dot_r64(n, dx, incx, dy, incy, result)
call blas_ddot_hpa_destroy()
```

  - Extra call before BLAS call to say use HPAs for subsequent calls to the named routine
  - Anchor and length of HPA specified
  - DDOT call made as before
    - Within library if an HPA has been initialized for this calculation the appropriate anchor and length will be used
  - Recommendation on destroying use of HPA after to avoid DDOT calls from subsequent routines (e.g. from LAPACK calls the user doesn't explicitly include themselves) having wrongly assumed HPA parameters
  - Note this example API is not thread safe – better may be subsuming all into a single call

Summary

- ARM Performance Libraries are the vendor supported BLAS and LAPACK implementation for AArch64
- Batched BLAS functions will appear once API standard is formalized
- Reduced Precision BLAS will be important for some sectors of our customers
  - Architecture support will be provided but we'll need to define correctness
- Reproducibility is something we are very keen to see on being supported through both our software and hardware

ARM