pt2pt wg
FP16
MPI Forum Meeting
Dec 4 - Dec 7, San Jose

Atsushi Hori
RIKEN AICS

Tickets

- #65 16-bit floating-point support for C/C++
  - DRAFT (will be updated soon after this meeting)
    https://github.com/mpi-forum/mpi-issues/files/1495174/FP16-Reading-1st-v2.pdf

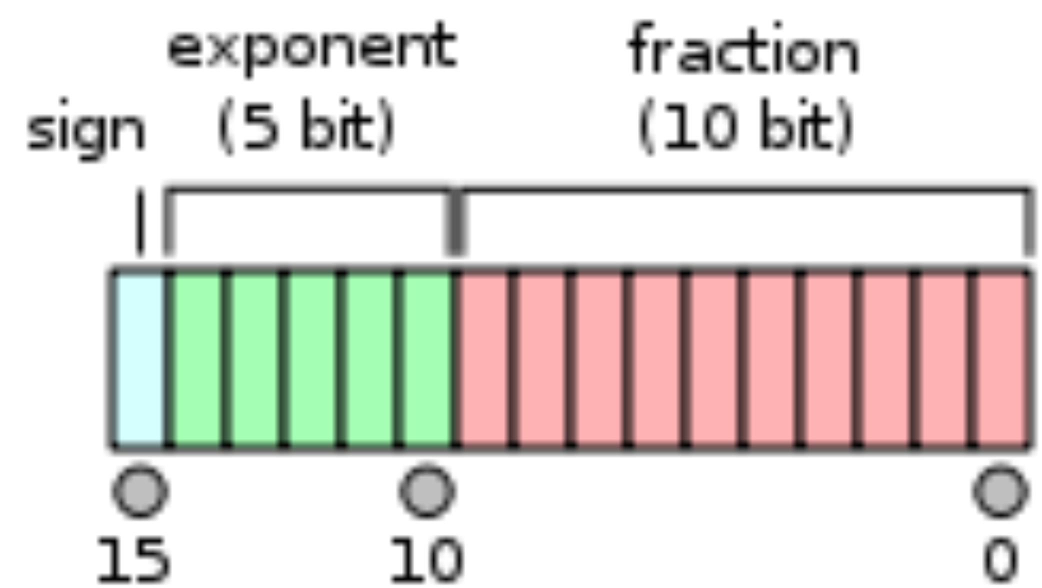
- #66 define language-agnostic, IEEE types
  - proposing to have not only BINARY*_T but also DECIMAL*_T types

- #69 Definition and Explanation of External32

What is “FP16” ?

- IEEE 754 2008 (ISO/IEC/IEEE 60559:2011)
  - FP16 was firstly introduced in 2008
  - The min. strictly positive (subnormal) value ≈ 5.96×10^{-8}.
  - The min. positive normal value ≈ 6.10×10^{-5}.
  - The max. representable value is = 65504.

- Current Status
  - “short float” on Xeon
    - no gcc 4.8.5
    - yes & no icc 18.0.1



ICC's "short float" is on the halfway

$ cat b.c
```
#include <stdio.h>
int main() {
  short float x = 1.0;
  x++;
  printf( "sizeof(x)=%d\n",
            sizeof(x) );
}
```

$ icc b.c

b.c(4): catastrophic error:
'short float' type unsupported
in this compiler version
compilation aborted for b.c
(code 1)

$ cat c.c
```
#include <stdio.h>
int main() {
  short float x = 1.0;
  // x++;
  printf( "sizeof(x)=%d\n",
            sizeof(x) );
}
```

$ icc c.c

$ ./a.out

sizeof(x)=2

FP16 in the standard

- **FP16 names**
  - **C/C++ Standarzation Proposal**
    - http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2016.pdf
    - http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0192r0.pdf

Now lets look at the lexical representation of floating types family names. Introduction of short float just made its lexical irregularity more obvious. What we mean is: while each of the floating types (sort of) acts as at int semantically, some floating point family **type names** do not look similar to the names of their cousins in integer's family at all:

```
short int    int    long int   long long int
short float  float  double     long double
```

5 New cstdfloat header

Similar to <cstdint> we propose adding new header <cstdfloat> with the following type defini- tions:

```
// architecture dependent, might be 32 bit
typedef short float        float16_t;
typedef float              float32_t;
typedef long float         float64_t;
// architecture dependent,
// might be 64 bit
typedef long long float    float128_t;
```

FP16 in C and C++

- http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2016.pdf

4.1 Implementation Options

As of storage and bit-layout for a short float number, we would expect most implementations to follow IEEE 754-2008 [11] half-precision floating point number format. On platform that do not provide any advantages of using shorter float, short float may be implemented as storage-only type, like `__fp16` on gcc/ARM today. For example, it can be stored in 'binary16' format in memory (occupying less bytes than float), converted to native 32-bit float on read from memory, operated on using native 32-bit float math operations and converted back to 'binary16' on store to memory. Or, the platform may choose to not take any advantages of `short float` and represent it using `float` in both memory and registers.

FP16 Support

- Possible cases of FP16 support outside of MPI
  1. Neither compiler nor CPU supports FP16
  2. Compiler and CPU support 2-byte FP16 natively
  3. Compiler produces codes with using up-scaled numbers -> SLOW

     load(16b), conv(16b->32b), compute(32b), conv(32b->16b), store(16b)
  4. Compiler treats FP16 as a synonym of FP32 (or higher) -> MORE MEMORY
- MPI FP16 support is OPTIONAL ?
  - if 3. or 4. is allowed (acceptable) then FP16 support can be mandatory
  - MPI-IO External32 defines the size of FP16

FP16 Global Sum

- FP16 global sum on huge array MAY produce less accurate results (than FP32 or higher).

- HQI may do the sum with the higher precision (behind the scene)
  - How much is enough ?

Revision Strategy

- Adding an *optional(?)* C type "short float"
  - MPI_SHORT_FLOAT (short float)
  - MPI_FLOAT16
  - MPI_FLOAT32
  - MPI_FLOAT64
  - MPI_FLOAT128
  - ~~MPI_SHORT_FLOAT_INT (reduction MIN_LOC and MAX_LOC)~~

stdfloat.h

Depends on RMA WG

- With Fortran, MPI_REAL2 and MPI_COMPLEX4 are defined already (see also slide 12)
- C/C++ interface *(optional?)*
  - MPI_C_SHORT_FLOAT_COMPLEX (short float _Complex)
  - MPI_CXX_SHORT_FLOAT_COMPLEX (std::complex<short float>)
  - although there is no explanation on these

MPI Standard X (X>3.1)

- CHAPTER 3.2. BLOCKING SEND AND RECEIVE OPERATIONS
  - Table 3.2: Predefined MPI datatypes corresponding to C datatypes
- CHAPTER 5.9. GLOBAL REDUCTION OPERATIONS
  - MIN_LOC, MAX_LOC
- CHAPTER 13.5. FILE INTEROPERABILITY
  - Table 13.2: "*external32*" sizes of predefined datatypes
- CHAPTER 17. LANGUAGE BINDINGS
  - Support for Size-specific MPI Datatypes
- ANNEX A. LANGUAGE BINDINGS SUMMARY
  - Table: Named Predefined Datatypes | C types
  - Table: Named Predefined Datatypes | C++ types
  - Table: Optional datatypes (Fortran) | Fortran types
- ANNEX A.1. DEFINED VALUES AND HANDLES
  - Table: Datatypes for reduction functions (C)
  - Table: Datatypes for reduction functions (Fortran)

Fortran REAL*2 and COMPLEX*4

- REAL*2 and COMPLEX*4 are already defined
  - Major Fortran compilers does not accept
    - gfortran, ifort and Cray fortran

- Should they be optional ?
  - MPI 3.1 Chap. 3.2.2 Message Data

> MPI requires support of these datatypes, which match the basic datatypes of Fortran and ISO C. Additional MPI datatypes should be provided if the host language has additional data types: MPI_DOUBLE_COMPLEX for double precision complex in Fortran declared to be of type `DOUBLE COMPLEX`; MPI_REAL2, MPI_REAL4, and MPI_REAL8 for Fortran reals, declared to be of type `REAL*2`, `REAL*4` and `REAL*8`, respectively; MPI_INTEGER1, MPI_INTEGER2, and MPI_INTEGER4 for Fortran integers, declared to be of type `INTEGER*1`, `INTEGER*2`, and `INTEGER*4`, respectively; etc.

Ticket #66: DECIMAL formats

- GCC supports decimal FP
  - https://gcc.gnu.org/onlinedocs/gcc/Decimal-Float.html
    - The decimal floating types are _Decimal32, _Decimal64, and _Decimal128.
    - GCC support of decimal float as specified by the draft technical report is incomplete:
      - When the value of a decimal floating type cannot be represented in the integer type to which it is being converted, the result is undefined rather than the result value specified by the draft technical report.
- ICC supports decimal FP
  - https://software.intel.com/en-us/node/523338
    - The Intel® C++ Compiler supports decimal floating point types in C and C++. The decimal floating point formats are defined in IEEE 754-2008 standard. In C, the following decimal floating types are supported: _Decimal32, _Decimal64, and _Decimal128.
- How many (HPC and/or MPI) applications use decimal FP ?

ticket #69: external32 (1/2)

- **In MPI 3.1**

  13.5.2 External Data Representation: "external32"

  All MPI implementations are required to support the data representation … All floating point values are in big-endian IEEE format [37] … Floating point values are represented by one of three IEEE formats. These are the IEEE "Single," "Double," and "Double Extended" formats, …

- **In IEEE Std 754™-2008**

  1 . O v e r v i e w

  1.1 Scope

  This standard specifies formats and methods for floating-point arithmetic in computer systems—standard and extended functions with single, double, extended, and extendable precision—and recommends formats for data interchange. Exception conditions are defined and standard handling of these conditions is specified.

ticket #69: external32 (2/2)

- IEEE Std 754™-2008

Table 3.5—Binary interchange format parameters

Parameter	binary16	binary32	binary64	binary128	binary{k} ($k \geq 128$)
k, storage width in bits	16	32	64	128	multiple of 32
p, precision in bits	11	24	53	113	$k - \mathrm{round}(4 \times \log 2\,(k)) + 13$
emax, maximum exponent e	15	127	1023	16383	$2^{(k-p-1)} - 1$
Encoding parameters					
bias, $E - e$	15	127	1023	16383	*emax*
sign bit	1	1	1	1	1
w, exponent field width in bits	5	8	11	15	$\mathrm{round}(4 \times \log 2\,(k)) - 13$
t, trailing significand field width in bits	10	23	52	112	$k - w - 1$
k, storage width in bits	16	32	64	128	$1 + w + t$

The function round( ) in Table 3.5 rounds to the nearest integer.

For example, binary256 would have $p = 237$ and $emax = 262143$.

- Changes in MPI X (X > 3.1)
  - These are the IEEE “binary16,” “binary32 (Single),” “binary64 (Double),” and “binary128” formats, …

Appendix

- https://en.wikipedia.org/wiki/IEEE_754

Name	Common name	Base	Significand Bits[a]/Digits	Decimal digits	Exponent bits	Decimal E max	Exponent bias[6]	E min	E max	Notes
binary16	Half precision	2	11	3.31	5	4.51	$2^4-1 = 15$	−14	+15	not basic
binary32	Single precision	2	24	7.22	8	38.23	$2^7-1 = 127$	−126	+127	
binary64	Double precision	2	53	15.95	11	307.95	$2^{10}-1 = 1023$	−1022	+1023	
binary128	Quadruple precision	2	113	34.02	15	4931.77	$2^{14}-1 = 16383$	−16382	+16383	
binary256	Octuple precision	2	237	71.34	19	78913.2	$2^{18}-1 = 262143$	−262142	+262143	not basic
decimal32		10	7	7	7.58	96	101	−95	+96	not basic
decimal64		10	16	16	9.58	384	398	−383	+384	
decimal128		10	34	34	13.58	6144	6176	−6143	+6144	