Julia on Fugaku

Mosè Giordano
m.giordano@ucl.ac.uk

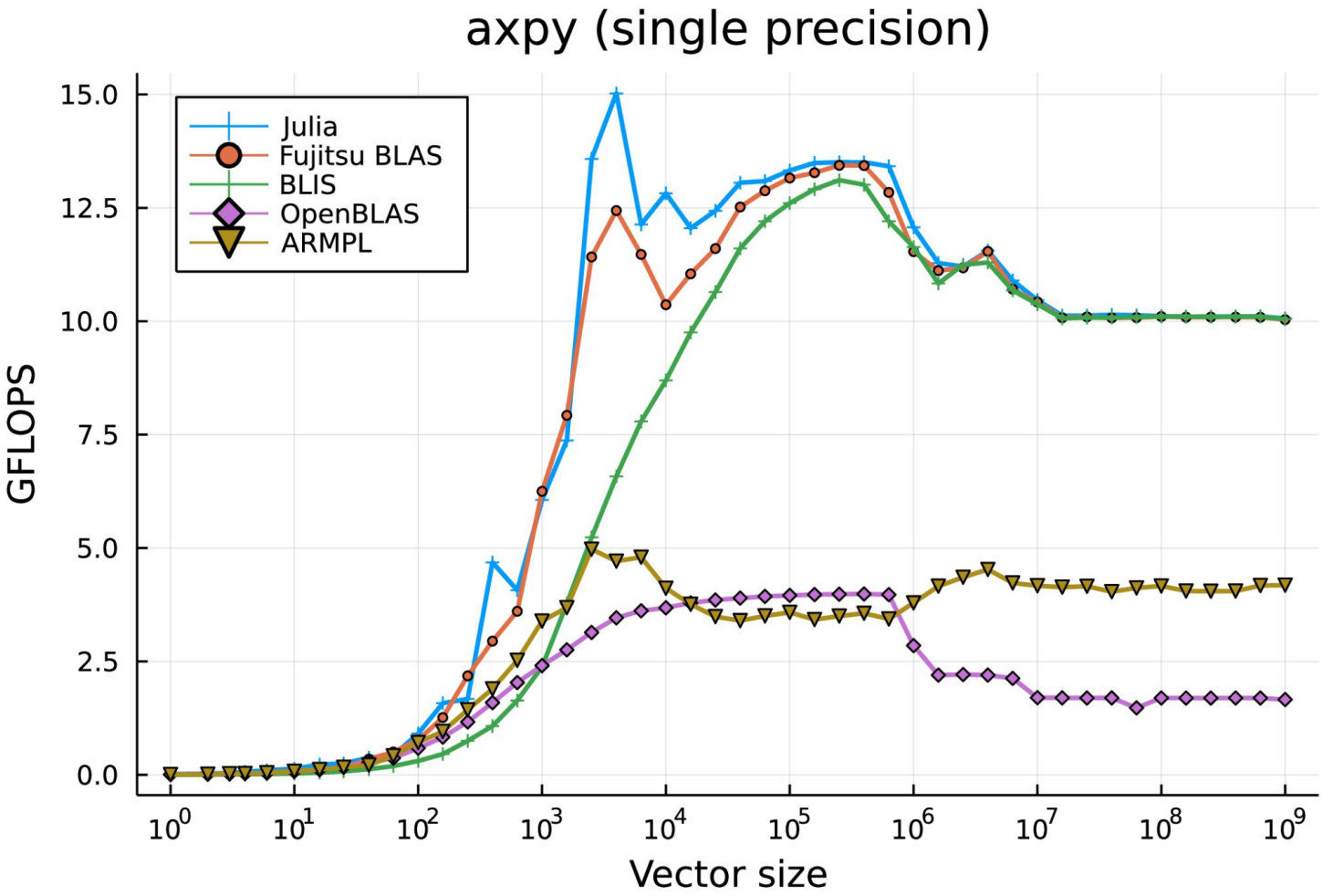
UCL
Centre for Advanced Research Computing

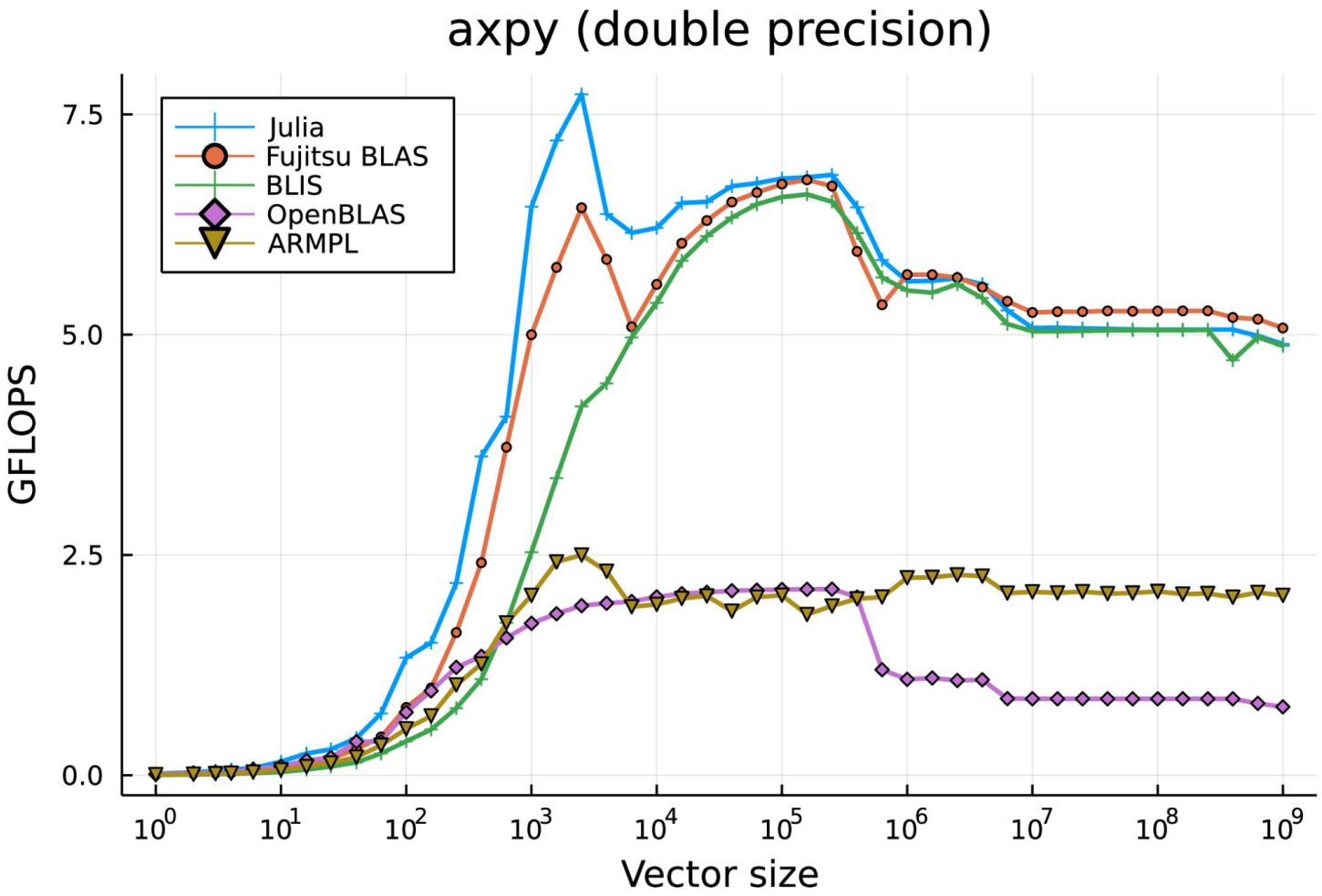
Level 1 BLAS showdown

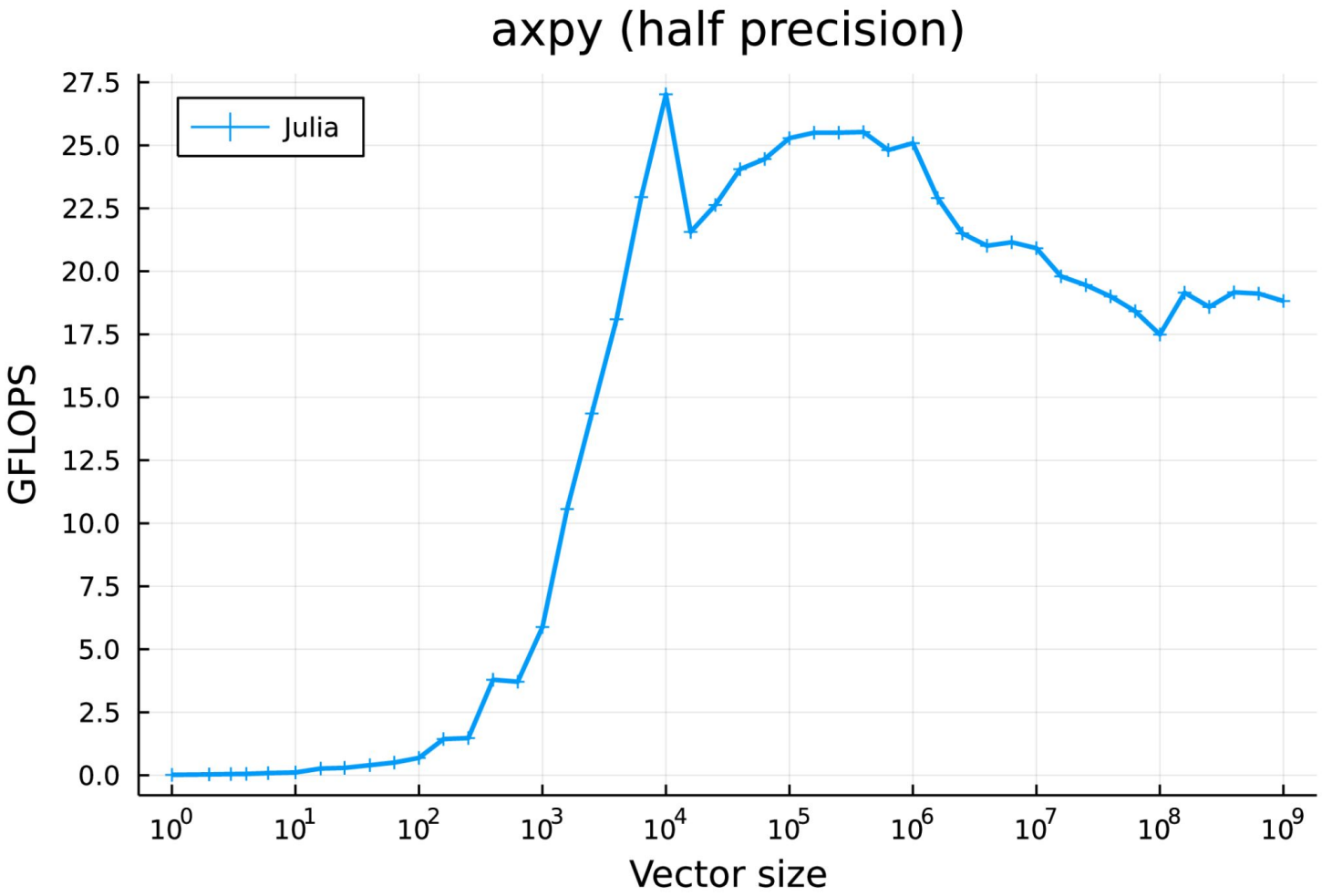
```
function axpy!(a, x, y)
    @simd for i in eachindex(x, y)
        @inbounds y[i] = muladd(a, x[i], y[i])
   end
   return y
end
```

vs

```
LinearAlgebra.BLAS.axpy!(a, x, y)
```


axpy (single precision)
Julia
Fujitsu BLAS
BLIS
OpenBLAS
ARMPL
GFLOPS
15.0
12.5
10.0
7.5
5.0
2.5
0.0
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Vector size


axpy (double precision)
Julia
Fujitsu BLAS
BLIS
OpenBLAS
ARMPL
GFLOPS
7.5
5.0
2.5
0.0
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Vector size

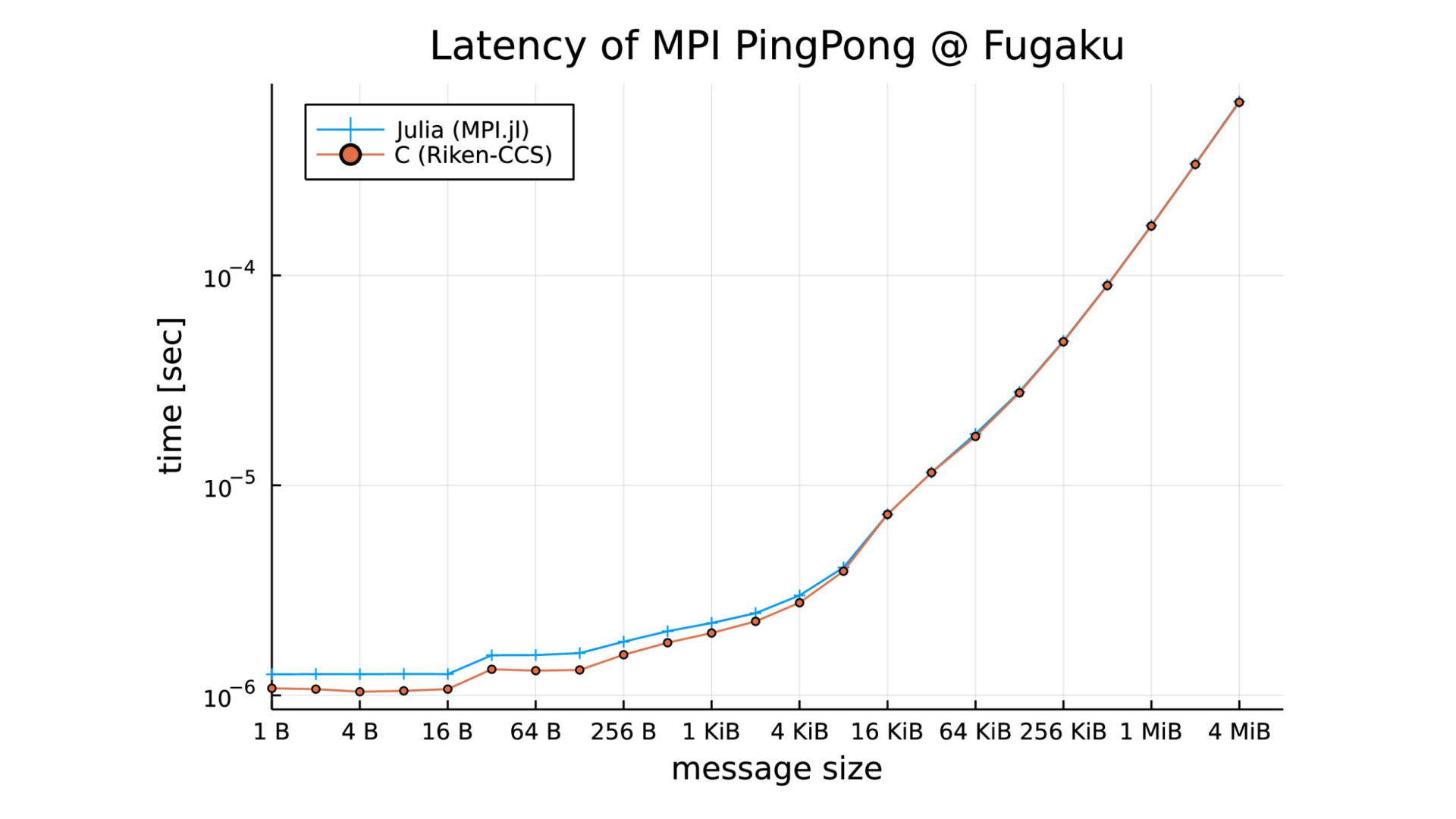

axpy (half precision)
Julia
GFLOPS
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Vector size

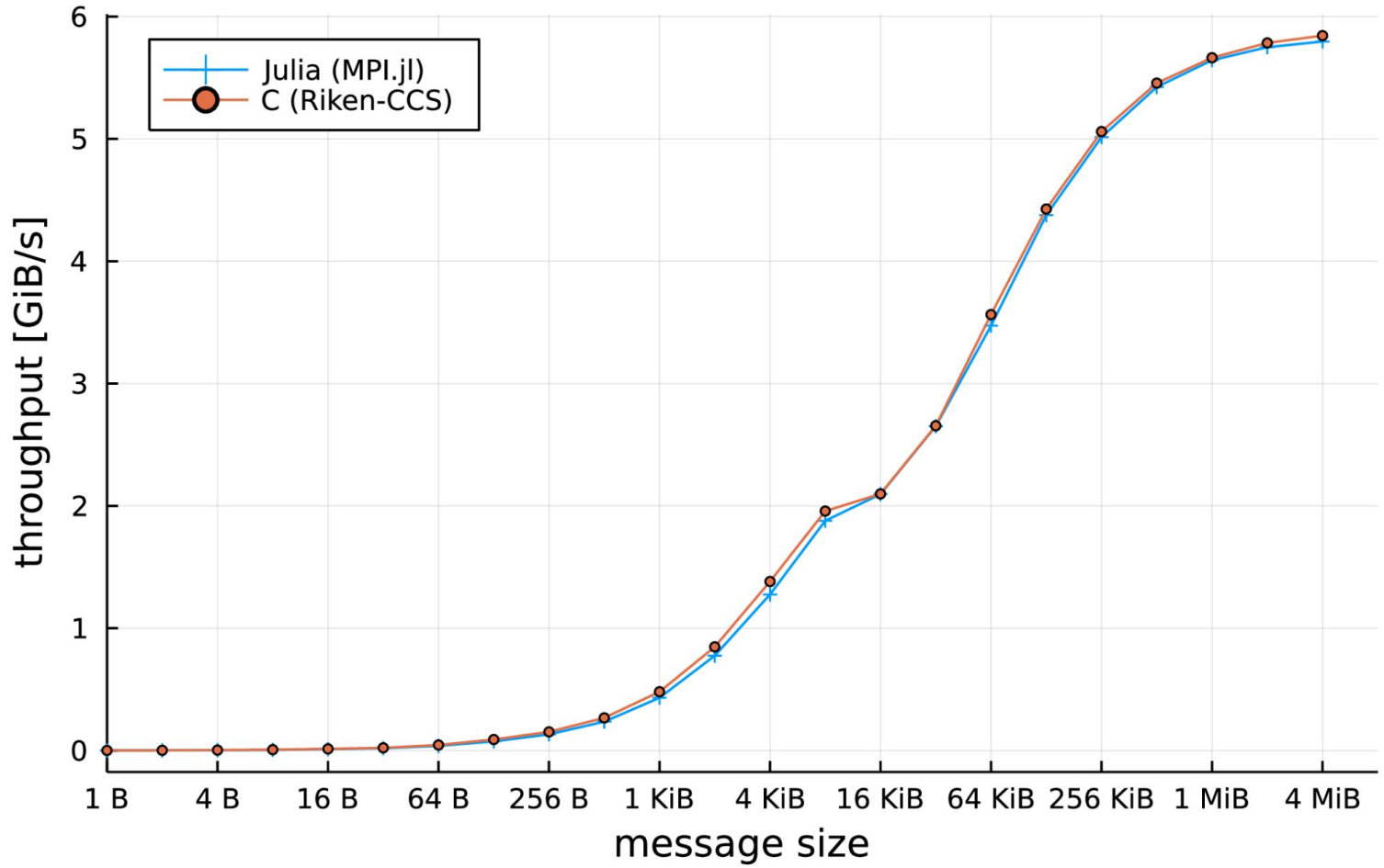
MPI.jl

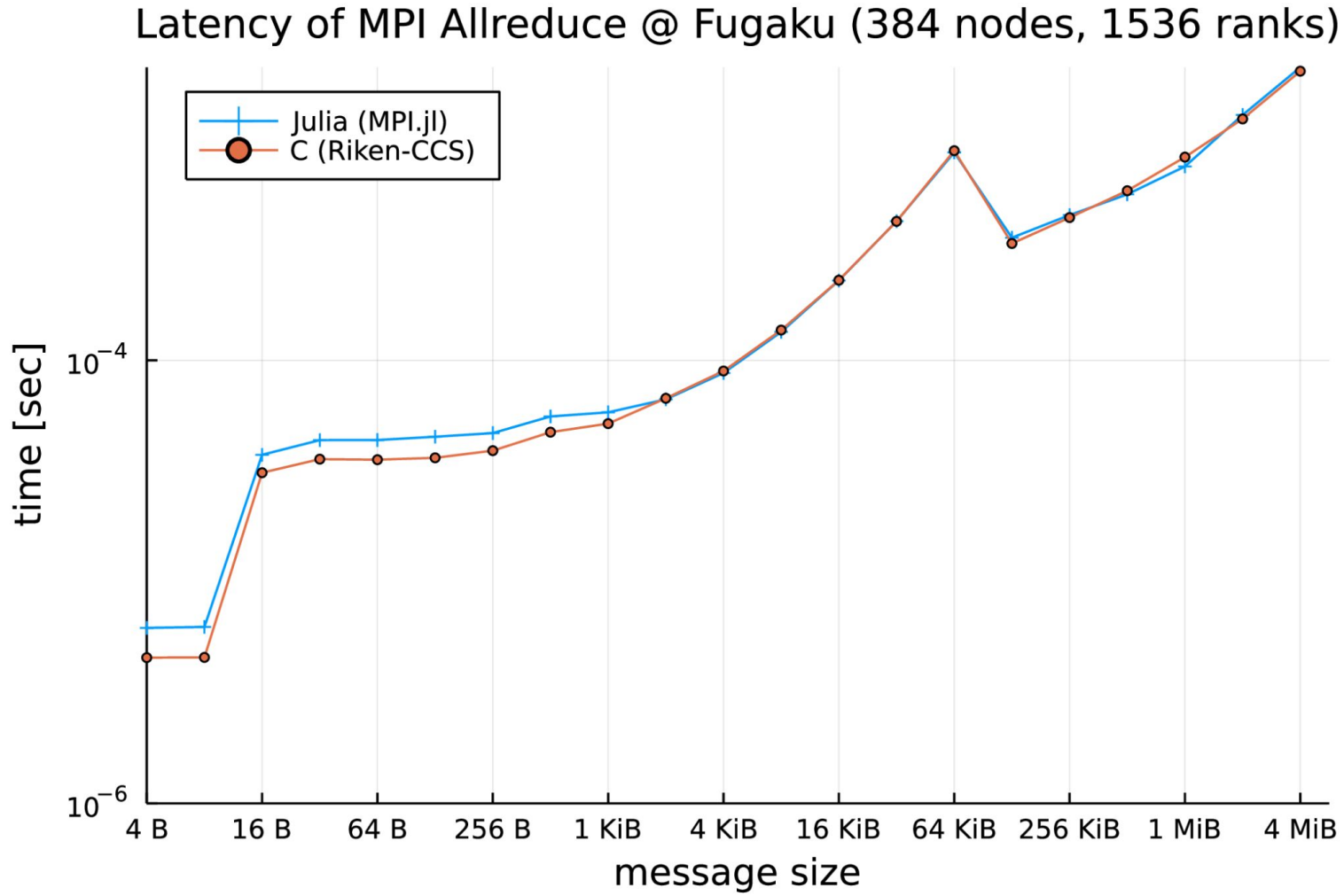
- Low-level access to MPI
- High-level convenience wrappers
- Deals with MPI ABI

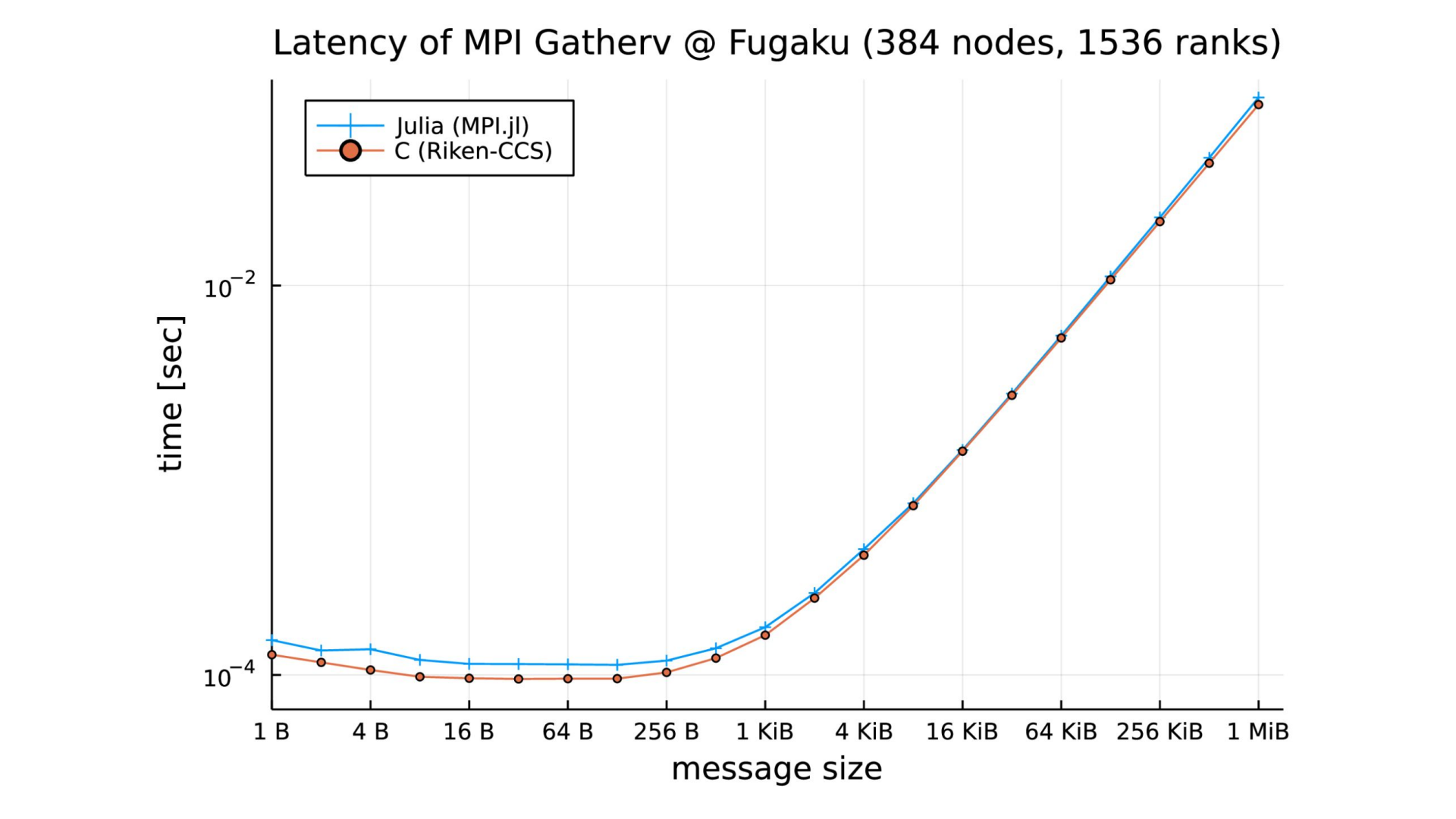
One of the oldest Julia packages (2012)

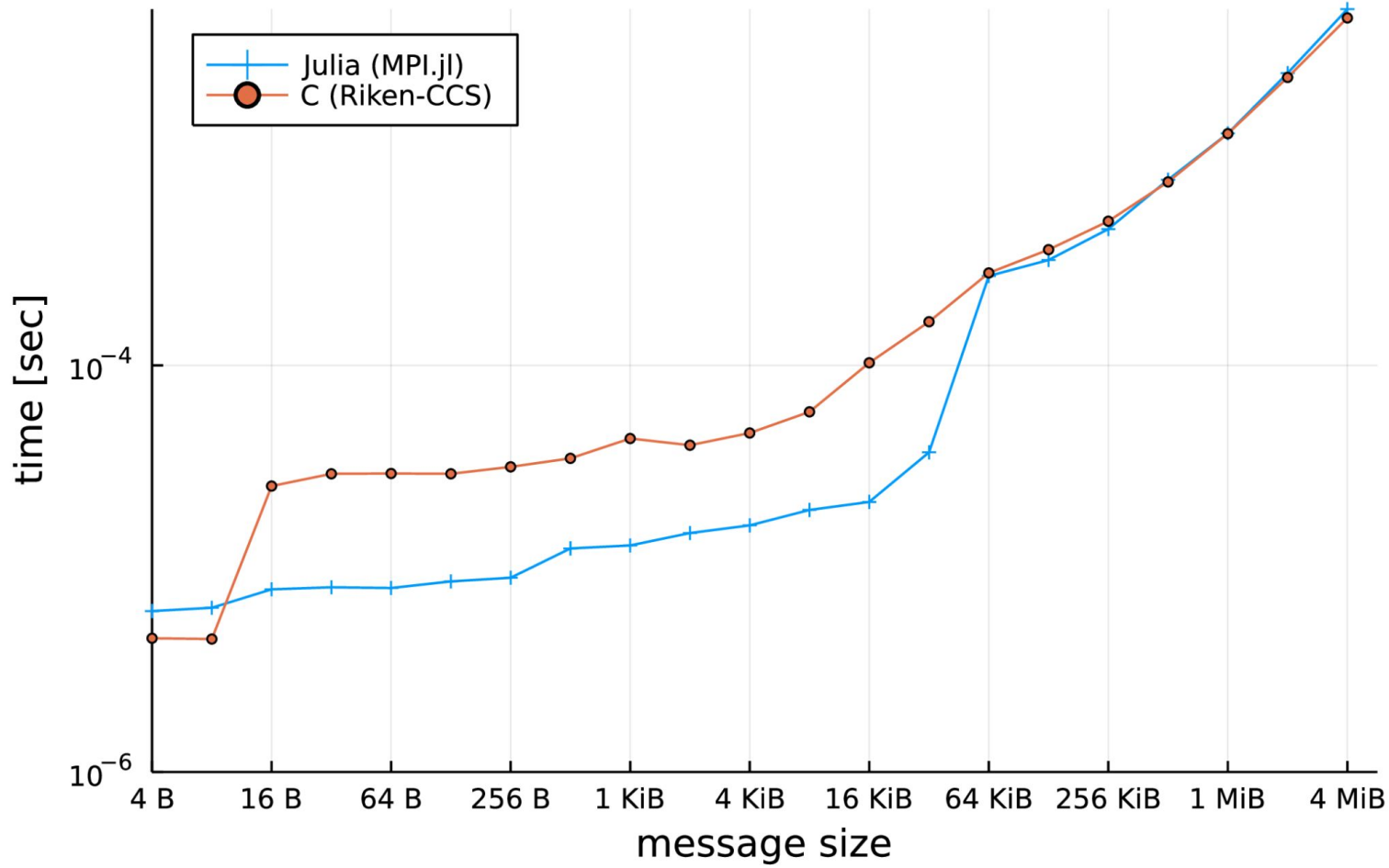
```
function pingpong(T::Type, bufsize::Int,
                  iters::Int, comm::MPI.Comm)
    rank = MPI.Comm_rank(comm)
    buffer = zeros(T, bufsize)
    tag = 0
    MPI.Barrier(comm)
    tic = MPI.Wtime()
    for i in 1:iters
        if iszero(rank)
            MPI.Send(buffer, comm; dest=1, tag)
            MPI.Recv!(buffer, comm; source=1, tag)
        elseif isone(rank)
            MPI.Recv!(buffer, comm; source=0, tag)
            MPI.Send(buffer, comm; dest=0, tag)
        end
    end
    toc = MPI.Wtime()
    return (toc - tic) / iters
end
```


Latency of MPI PingPong @ Fugaku
Julia (MPI.jl)
C (Riken-CCS)
time [sec]
10^{-4}
10^{-5}
10^{-6}
1 B
4 B
16 B
64 B
256 B
1 KiB
4 KiB
16 KiB
64 KiB
256 KiB
1 MiB
4 MiB
message size


Throughput of MPI PingPong @ Fugaku
Julia (MPI.jl)
C (Riken-CCS)
throughput [GiB/s]
0
1
2
3
4
5
6
1 B
4 B
16 B
64 B
256 B
1 KiB
4 KiB
16 KiB
64 KiB
256 KiB
1 MiB
4 MiB
message size


Latency of MPI Allreduce @ Fugaku (384 nodes, 1536 ranks)
Julia (MPI.jl)
C (Riken-CCS)
time [sec]
10^{-4}
10^{-6}
4 B
16 B
64 B
256 B
1 KiB
4 KiB
16 KiB
64 KiB
256 KiB
1 MiB
4 MiB
message size


Latency of MPI Gatherv @ Fugaku (384 nodes, 1536 ranks)
Julia (MPI.jl)
C (Riken-CCS)
time [sec]
10^{-2}
10^{-4}
1 B
4 B
16 B
64 B
256 B
1 KiB
4 KiB
16 KiB
64 KiB
256 KiB
1 MiB
message size


Latency of MPI Reduce @ Fugaku (384 nodes, 1536 ranks)
Julia (MPI.jl)
C (Riken-CCS)
time [sec]
10^{-4}
10^{-6}
4 B
16 B
64 B
256 B
1 KiB
4 KiB
16 KiB
64 KiB
256 KiB
1 MiB
4 MiB
message size

Opportunities for improvements

- Julia is not optimised for A64FX, but every version gets better thanks to upstream improvements in LLVM
  - Keep them coming!
- Compilation latency on A64FX hits particularly hard a JIT language
  - There are ongoing works to continue reducing compilation latency in Julia and to improve static compilation story
- Custom reductions don’t work in MPI.jl on non-Intel architectures
  - Can be fixed, but it needs someone to do the work
- ~~Runtime detection of hardware support for Float16~~ Done!
  - Recently tested also on AVX512-FP16