IOWA STATE UNIVERSITY

EuroMPI/USA 2017
September 25-28, 2017

Nathan T. Weeks[1,2], Meiyue Shao[3], Brandon Cook[4], Marcus Wagner[5], Glenn R. Luecke[2], Pieter Maris[6], James P. Vary[6]

# Accelerating MPI Reductions on Intel Xeon Phi

## Introduction

Since the introduction of MPI global reduction operations in the first MPI standard, the possible memory footprint per MPI process has grown, resulting in the ability to perform reductions on larger arrays. Compared to preceding multi-core processors, the Intel Xeon Phi "Knights Landing" (KNL) many-core processor presents different performance opportunities (more cores/threads; wider vector widths) and pitfalls (lower serial performance) affecting reduction operations.

## Objective

Improve MPI sum reduction performance on KNL using compile-time optimizations (including OpenMP).

## Methodology

- Establish baseline performance for predefined MPI_SUM reduction operation on $2^{30}$ REALs (single-precision float-point number) per rank
- Compare MPI_SUM with equivalent user-defined reduction operation with AVX-512 vectorization
- Add threading to user-defined reduction operation (OpenMP taskloop)

## Benchmark Environment

- NERSC Cori (Intel Xeon Phi 7250, 68 cores; *cache* memory mode; *quadrant* clustering mode)
- Default Cori software environment, substituting cray-mpich/7.5.5, Intel Fortran 17.0.3, and craype-mic-knl
- Source code and results available at: https://doi.org/10.5281/zenodo.827377

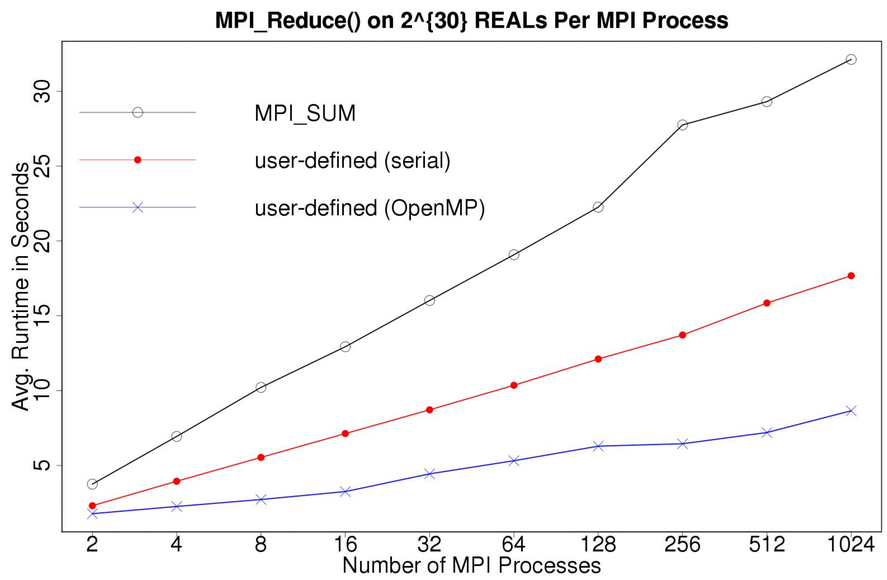


**Figure 1.** Timings for predefined (MPI_SUM) and user-defined (serial and OpenMP) sum reductions on $2^{30}$ REALs per MPI process (1 MPI process per KNL node; 64 threads for the OpenMP version). Each reported timing is the average of 10 calls to MPI_Reduce().

## Results

- ~1.8X speedup from suitably-vectorized user-defined reduction operation equivalent to predefined MPI_SUM
- Additional ~2X speedup using multithreading (OpenMP taskloop)

## Conclusions

Codes that perform many MPI sum reductions on large arrays on KNL may benefit from use of simple multi-threaded user-defined reductions.

1. Department of Computer Science, Iowa State University
2. Department of Mathematics, Iowa State University
3. Computational Research Division, Lawrence Berkeley National Laboratory
4. National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory
5. Cray, Inc.
6. Department of Physics, Iowa State University

```
! Multi-threaded user-defined equivalent to MPI_SUM
subroutine openmp_sum(invec, inoutvec, len, datatype)
   integer, intent(in)    :: len, datatype
   real,    intent(in)    :: invec(len)
   real,    intent(inout) :: inoutvec(len)
   integer                :: i
!$omp taskloop simd default(shared) priority(1)
   do i = 1, len
      inoutvec(i) = inoutvec(i) + invec(i)
   end do
end subroutine openmp_sum
! Potential application of taskloops to facilitate
! concurrent computation with MPI reduction (not
! reflected in benchmark)
program main
…
   call MPI_Op_create(user_sum, .true., op, ierror)
…
!$omp parallel
!$omp master
!$omp task priority(1)
   call MPI_Reduce(sendbuf, recvbuf, count, MPI_REAL,&
                   op, comm, root, ierror)
!$omp end task
!$omp taskloop priority(0)
   do …
   … other tasks executing concurrently with reduction
   end do
! ensure MPI_Reduce task done before reusing buffers
!$omp taskwait
…
```

**Figure 2.** Conceptual application for OpenMP taskloop in user-defined MPI reduction, allowing load balancing with concurrent tasks/taskloops outside of reduction. Tasks generated by the reduction taskloop are prioritized.

**Acknowledgements** This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.