D4.5.4 – Non-blocking collectives in one or more of the co-design applications

WP4: Algorithms and Libraries

Project Acronym	CRESTA
Project Title	Collaborative Research Into Exascale, Systemware, Tools and Applications
Project Number	287703
Instrument	Collaborative project
Thematic Priority	ICT-2011.9.13 Exascale computing, software and simulation

Due date:	M39
Submission date:	31/12/2014
Project start date:	01/10/2011
Project duration:	39 months
Deliverable lead organization	HLRS
Version:	1.0
Status	Internal Review
Author(s):	Pekka Manninen (Cray), Rupert W. Nash (UEDIN), Christoph Niethammer (HLRS), Dmitry Khabi (HLRS)
Reviewer(s)	Dan Henningson (KTH) Tobias Hilbrich (TUD)

Dissemination level	
PU	*PU - Public*

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	10/11/2014	First draft	Dmitry Khabi(HLRS)
0.2	15/11/2014	Section about “Overlap availability of non-blocking collectives”	Pekka Manninen(Cray)
0.3	11/27/2014	Section about “Impact of the entering time on collective performance”	Christoph Niethammer (HLRS)
0.4	11/28/2014	Sections about CG and HemeLB	Dmitry Khabi(HLRS)
0.5	05.12.2014	Addressing comments	Pekka Manninen(Cray)
0.6	05.12.2014	Addressing comments	Dmitry Khabi(HLRS)
0.7	11.12.2014	Addressing comments	Christoph Niethammer (HLRS)
1.0	16.12.2014	Final version for submission	Catherine Inglis (UEDIN)

Table of Contents

Index of Figures

1 Executive Summary

Deliverable 4.3.2 contains a description of the work done under CRESTA in the field of the non-blocking collectives. The experiences and insights in this field have been incorporated in the two papers “Benchmarking MPI Collectives” (1) and “MPI collectives at scale” (2), which were presented at Supercomputing Conference 2014.

The objective of this document is to collect key information about non-blocking collectives and their possible use in High Performance Computing. Detailed information can be obtained in the above-mentioned papers.

In sections 3 and 4 we study the influence of the entering time on some existing collective operations and the possibility of hiding the parallel overhead caused by the operation by overlapping the communication with computation or other work.

The information about the usage of the non-blocking collective operation MPI_Iallreduce in the iterative solvers of NekBone (3) and CEL library (4) is provided in section 5.

Section 6 contains information about the integration of non-blocking collectives into a production application, HemeLB (5).

2 Introduction

The MPI 3.0 (6) standard introduced non-blocking collective operations which give new optimization opportunities for applications, since they allow overlap of communication with computation, thus reducing synchronisation costs for delayed processes. In our research we take into account not only the benchmark results for blocking and non-blocking collective operations, but also their performance impact on real world applications. To explore the performance consequences of using non-blocking collectives in a production application, we have implemented an alternative version of the global monitoring aspects of HemeLB and of the reduction operations in the iterative solvers of the NekBone and CEL solver library.

3 Impact of the entering time on collective performance

There are a lot of different collective algorithms around in the area of HPC: barriers, reductions, scatter, gather or alltoall to just mention a few of them. Many of them are extremely important for applications and are the limiting factor when it comes to scalability. While they are frequently used in HPC applications, and optimized algorithm implementations exist for different numbers of PEs, hardware architectures and networks, they are mostly based on the assumption of a synchronized start and a more or less homogeneous communication system. While the influence of noise on blocking collective operations has already been studied (7), little is known about the detailed influence of different entering times of processes e.g. caused by load imbalances, especially on non-blocking collectives.

Within CRESTA we did a first study on the influence of entry time on a set of existing collective operations within MPI. For this, a new testing method was developed and implemented as a benchmarking suite.

The testing approach includes the best effort synchronization at the beginning of each test. After this, all but one process enter the collective to be studied. The one process is delayed by a given time before entering the collective (see Fig. 1).

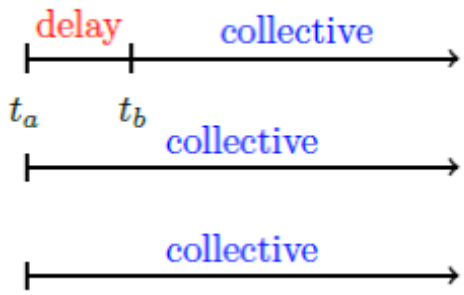

Fig. 1: Processes are synchronized at time t_a and all but one enter the collective. The delayed process enters the collective at time $t_b = t_a + \delta$.

The following collectives were studied in detail: MPI_Barrier, MPI_Allreduce and MPI_Alltoall, as well as their non-blocking counterparts. Benchmarks were run on the Hermit system at HLRS, which uses the 3D-torus network Gemini. The results for the MPI implementation there show a small benefit for delays of 10, 20 and 50 micro seconds up to 1024 processes – the delay is partially overlapped with the collective operation here. A negative effect can be detected for larger number of processes. For more details we refer to the detailed description in (1).

4 Overlap availability of non-blocking collectives

One key argument for introducing non-blocking collectives is the possibility of hiding the parallel overhead caused by the operation by overlapping the communication with computation or other work. The usage pattern would then be:

- initialize the collective operation;
- do work not requiring the data involved in the communication; and,
- wait for the collective to finish.

Not all algorithms have this independent work available for the overlap and, even if they do, it depends on the implementation of the MPI library whether the non-blocking communication actually happens simultaneously with the overlapped work, or whether it occurs in the waiting phase only.

We can benchmark how well an MPI library is able to overlap collectives as follows:
1. Measure the average time over all MPI ranks needed to perform the non-blocking collective operation (T_{coll});
2. Measure the average time over all MPI ranks needed to perform a matrix-vector multiplication of size equal to the number of MPI ranks (T_{comp})
3. Measure the time needed for the above combined and overlapped operations ($T_{overlap}$).

Then, the execution time saved by performing the overlap is

$$T_B = \left(T_{coll} + T_{comp}\right) - T_{overlap}$$

and the relative benefit

$$\beta = 100\% \cdot \frac{T_B}{T_{overlap}}$$

The value 1 (100%) represents an ideal case, where the communication overhead can be hidden altogether, and negative values imply that the overlap in fact slows down the overall execution, and hence it would be better not to overlap at all (compare with "delay overlap benefit" introduced in Section 3). The benchmark can be found from the CRESTA Collective Communication Library (8). We measured the relative benefit from the overlap of the all-to-all data exchange (MPI_Ialltoall) and the global reduction (MPI_Iallreduce) operations, which are typical bottleneck collectives in many HPC applications. The measurements were carried out in the Cray XC30 architecture, and reported in the CRESTA whitepaper “Benchmarking MPI collectives” (1). Our findings were that the computation-communication overlap is not always available, but depending on the platform, operation, size of the communicator, and the amount of data to be communicated, some performance gains may be obtained by performing the overlap; but the programmer should also verify that performing the overlap is not causing performance degradation.

5 Non-blocking collectives in the iterative linear solvers

One iteration of a typical iterative solver consists of a sparse-matrix vector multiplication and various vector operations, including the collective operation *allreduce*. The MPI_Allreduce operation was replaced with the two operations MPI_Iallreduce and MPI_Wait. In this section we show the performance of the non-blocking and blocking versions of the iterative solvers in NekBone and the CEL solver libraries.

The right part of Figure 1 shows the performance of the CG algorithm on a CrayXE6 with blocking MPI_Allreduce and non-blocking MPI_Iallreduce operations. The two curves show very similar behavior. There are two cases where the performance shows slight variation: 256 and 512 cores. The left part of the figure shows the Cray-Pat profiling for 256 and 512 cores. The diagrams show that the collectives MPI_Iallreduce and MPI_Allreduce have different implementations. This difference and the slight load imbalance of the calculation influence the performance. The performance may be better or worse than anticipated. That has also been shown by the previously-described collective benchmarks (see (1) and (2) for more details).

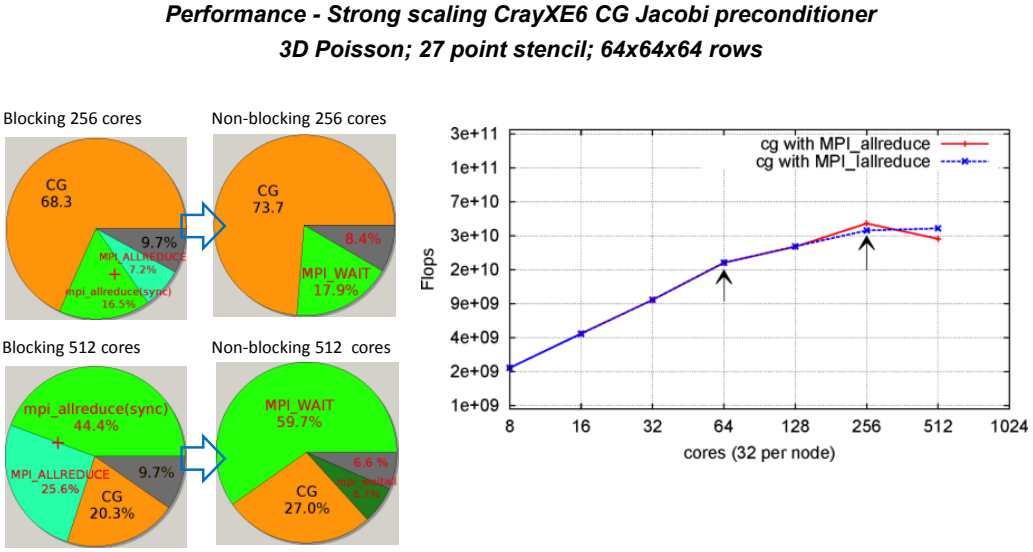

Figure 1 - Comparison between CG implementations with non-blocking and blocking collective operation *allreduce*.

A standalone benchmark version of the full Nek5000 application, called NekBone, was also used for the integration of the non-blocking collective operation MPI_Iallreduce. Figure 2 shows a small difference in the runtime between the blocking and non-blocking version of the CG in NekBone application.
Unfortunately, there are still no clear parameters on which to base the decision about which of the two operations is better to use in different situations.

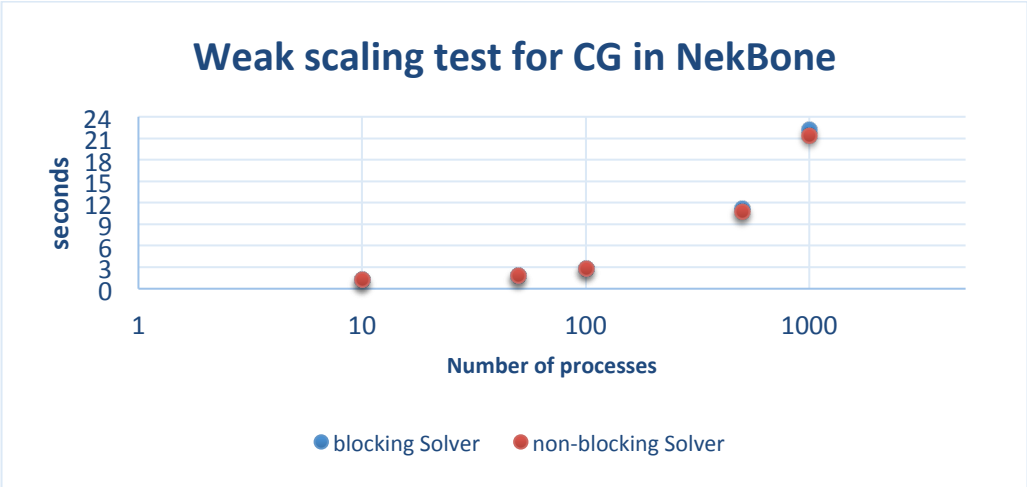

Figure 2 - Runtime of 100 iterations of non-blocking and blocking verisons of NekBone CG on Milner (Cray XC30) (9). This test and integration of non-blocking collectives into NekBone was done by Dana Akhmetova (KTH).

6 Non-blocking collectives in the production application HemeLB

HemeLB (10) is a lattice-Boltzmann based fluids solver, optimised for simulation of blood flow in domains derived from 3D angiography data. Previous work has shown that its computational performance scales linearly up to at least 32,768 cores (5) on HECToR, the UK's previous-generation national supercomputer.

The core lattice-Boltzmann algorithm requires data exchange between neighbouring points only, giving very high potential scalability. Further, HemeLB updates the sites on inter-rank boundaries at the start of the timestep and begins communicating the necessary data, before proceeding to update those sites that do not need data from another rank. The code then waits for communication to finish and updates the boundary sites.

The phased-communicator uses non-blocking point-to-point MPI operations, which are posted and waited on at the same time as the core lattice-Boltzmann communications. This keeps the performance impact of the global monitoring very low, but comes at the price of significant software complexity and adds a multiple timestep delay until the result is known. As a proof of concept, we have replaced this phased communication with a lightweight wrapper around MPI 3.0 asynchronous collectives.

The implementation significantly reduced the code complexity of the affected components without significantly changing the performance, despite allowing significantly more frequent monitoring of global quantities (see Figure 3). This also holds true for the results obtained with both benchmarks considered above.

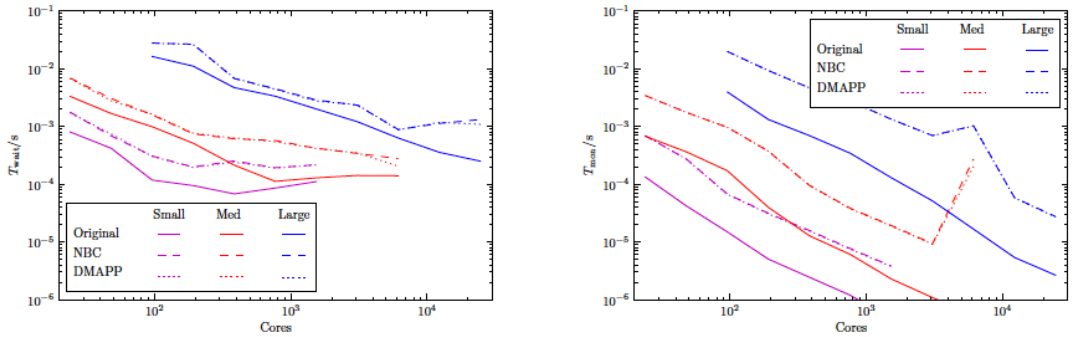

Figure 3 - Left: time spent in MPI_Wait calls. Right: time spent in monitoring calculations. The line colour indicates the size of problem used (small: magenta; medium: red; large: blue) and the style of line indicates the type of collectives used (default: solid; NBC: dashed; NBC with DMAPP: dotted).

For more details we refer to the detailed description in (1).

7 Outlook

The non-blocking collectives were considered not only using synthetic benchmarks, but also in an already-optimized production application, HemeLB. Although our performance measurement has shown that the integration of non-blocking collectives does not significantly change performance, the usage of the non-blocking collectives has significantly simplified the monitoring code of HemeLB.

This work shows that the state-of-the-art implementation of the non-blocking collectives in Cray MPI is as good or better than their blocking counterparts - in benchmarks and real world applications. As the specification of this MPI 3.0 interface is relatively new, we expect new algorithms with better overlapping capabilities and hardware with even better support for offloading communication for the future. The techniques for overlapping communication may also improve collective operations in the case of late arrivals. Our preliminary work in this area already shows some potential to hide small delays of single processes for the barrier, all-reduce and all-to-all operations.

8 Bibliography

1. White Paper Benchmarking MPI Collectives at SC14. Christoph Niethammer, Pekka Manninen, Rupert Nash, Dmitry Khabi, Jose Gracia. s.l. : CRESTA Consortium Partners, 2014.

2. MPI collectives at scale. Christoph Niethammer, Pekka Manninen, Rupert W. Nash, Dmitry Khabi, Jose Gracia. Workshop on Exascale MPI at Supercomputing Conference 2014 : s.n., 2014.

3. Nek5000 project web page. [Online] http://nek5000.mcs.anl.gov/.

4. D4.3.2 Community prototype of exascale algorithms and solver (Software). Dmitry Khabi, Frederic Magoules. s.l. : CRESTA Consortium Partners 2011, 2014.

5. Analysing and modelling the performance of the HemeLB lattice-Boltzmann simulation environment. Derek Groen, James Hetherington, Hywel B Carver, Nash, Rupert W Nash, Miguel O Bernabeu, and Peter V. Coveney. J. Comput. Sci. 4, 2012, Vols. p. 412--422, 5.

6. MPI: A Message-Passing Interface Standard Version 3.0 Chapter author for Collective Communication, Process Topologies, and One Sided. s.l. : Message Passing Interface Forum. , Sep. 2012.

7. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. T. Hoefler, T. Schneider, and A. Lumsdaine. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10). 2010.

8. D4.5.3 Non-Blocking Collectives Runtime Library. Pekka Manninen.. s.l. : CRESTA Consortium Partners 2011, 2013.

9. The compute rack of PDC's supercomputer Milner. [Online] [Cited:] https://www.pdc.kth.se/resources/computers/milner.

10. Choice of boundary condition for lattice-Boltzmann simulation of moderate-Reynolds-number flow in complex domains. R. W. Nash, H. B. Carver, M. O. Bernabeu, J. Hetherington, D. Groen. Phys. Rev. E, vol. 89, p. 023303. 2014.

11. D4.5.2 Microbenchmark Suite. José Gracia, Christoph Niethammer, Wahaj Sethi. s.l. : CRESTA Consortium Partners, 2012.

12. Cray XE6 (HERMIT). [Online] 2014. [Cited: 30 10 2014.] https://wickie.hlrs.de/platforms/index.php/Cray_XE6.

13. CRAY XC40 (HORNET). [Online] HLRS, 2014. [Cited: 30 10 2014.] http://www.hlrs.de/systems/platforms/cray-xc40-hornet/.

14. Analysing and modelling the performance of the HemeLB lattice-Boltzmann simulation environment. D. Groen, J. Hetherington, H. B. Carver, R. W. Nash, M. O. Bernabeu, P. V. Coveney. J. Comput. Sci. sep. 2012, Vol. 4, no. 5, pp. 412–422.

15. Coalesced communication: a design pattern for complex parallel scientific software. H. B. Carver, D. Groen, J. Hetherington, R. W. Nash, M. O. Bernabeu, and P. V. Coveney. submitted to Advances in Engineering Software : http://arxiv.org/abs/1210.4400v1, 2014.