Performance improvement of MODYLAS using Remote Direct Memory Access on the K computer

Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato
RIKEN R-CCS
Kobe, Hyogo, Japan

Yoshimichi Andoh, Susumu Okazaki
Nagoya University
Nagoya, Aichi, Japan

ABSTRACT

We have developed the general-purpose molecular dynamics software MODYLAS, which is executed on large-scale supercomputers. In order to improve its strong scalability, this research replaces MPI communication with Remote Direct Memory Access (RDMA) on the K computer. Since the K computer provides the extended RDMA interface for RDMA operations, we implement a library to use the interface easily from MODYLAS. When measuring performance of MODYLAS, the RDMA communication time is 29~42% less than the MPI communication time.

CCS CONCEPTS

• **General and reference** → *Evaluation.*

KEYWORDS

molecular dynamics, fast multipole method, RDMA

ACM Reference Format:
Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato and Yoshimichi Andoh, Susumu Okazaki. 2019. Performance improvement of MODYLAS using Remote Direct Memory Access on the K computer. In *ICPP 2019: ACM Symposium on Neural Gaze Detection, August 05–08, 2019, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Molecular dynamics (MD) calculations are widely used as an analysis tool in various fields such as chemistry, physics, biology, virology, and engineering. However, the time that can be simulated by MD calculation in practical research is limited to around $10^{-6} \sim 10^{-5}$ seconds for a 10^7 atomic system even when using a state-of-the-art supercomputer. Breakthroughs in the above fields can be expected if it becomes possible to perform long-term and larger-scale MD calculations.

We have developed the general-purpose molecular dynamics software MODYLAS[1], which utilizes the fast multipole method (FMM) for the calculation of electrostatic interactions. Our preliminary evaluation indicates the time required for MPI communication is limited by the communication latency. This tendency is particularly noticeable under strong scaling. Therefore, in order to reduce the communication latency, this research replaces MPI communication for the atomic coordinates and multipole coefficients in MODYLAS with Remote Direct Memory Access (RDMA) communication on the K computer and evaluates the performance of our proposed changes.

ICPP 2019, August 05–08, 2019, Kyoto, Japan

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

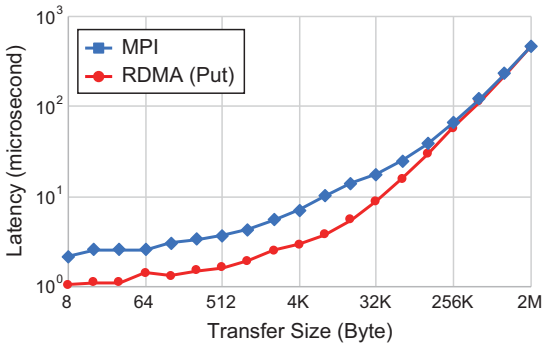

Figure 1: Communication latency on the K computer

2 IMPLEMENTATION

2.1 MODYLAS

MODYLAS can perform large-scale molecular dynamics simulations on large-scale supercomputers such as the K computer (CPU: SPARC64 VIIIfx 2GHz, Memory: DDR3 SDRAM 16GB, Network: Torus fusion interconnect 5GB/s)[2]. MODYLAS is written in Fortran and its source code can be obtained from the official website (https://www.modylas.org). Since communications in MODYLAS are designed so that partners are limited to adjacent processes, the MPI functions mpi_isend(), mpi_irecv(), and mpi_wait() are used.

2.2 RDMA on the K computer

The K computer provides users with the extended RDMA interface[3] so that they can issue RDMA operations (Put/Get) with low latency in addition to MPI communication. To measure its performance, we implement a pingpong benchmark using the interface. Fig. 1 shows a comparison of the latency between MPI and RDMA (Put) on the K computer. This result indicates that the performance of RDMA (Put) is always better than that of MPI.

2.3 Replacement of MPI with RDMA

We implement a library to use the extended RDMA interface from MODYLAS easily because the current interface is very primitive and written in C language. Fig. 2 shows a part of the code using the library. This code uses macros to switch between MPI and RDMA. The prefix for the subroutines provided by the library is "rdma_."

In line 4, the subroutine "rdma_register_addr" targets the specified array "ircbufp" for RDMA and obtains the remote addresses of

```
1  integer(4),allocatable,dimension(:) :: icbufp
2  allocate(ircbufp(max_cell*nbd))
3  #ifdef RDMA
4  call rdma_register_addr(ircbufp, (max_cell*nbd)*4)
5  #endif
6     :
7  #ifdef RDMA
8  integer(8),pointer :: ircbufp_raddr(:)
9  type(c_ptr) :: ircbufp_cptr
10 ircbufp_cptr = rdma_get_raddr(ircbufp)
11 call c_f_pointer(ircbufp_cptr, fptr=ircbufp_raddr, shape=[nprocs])
12 call rdma_put_post(ipz_pdest, ircbufp_raddr(ipz_pdest+1), ...)
13 call rdma_wait(ipz_psrc)
14 #else
15 call mpi_irecv(ircbufp, ..., ipz_psrc, ...)
16 call mpi_isend(icbufp, ..., ipz_pdest, ...)
17 call mpi_waitall(2, ...)
18 #endif
```

Figure 2: Modified code of MODYLAS

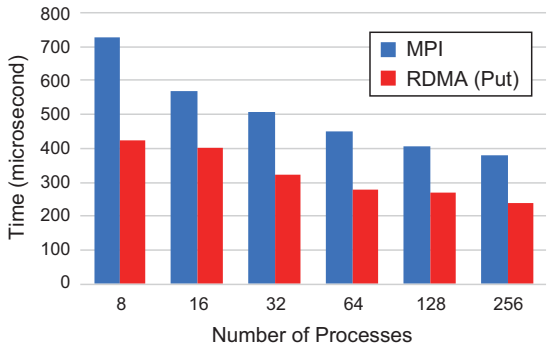

Figure 3: Communication time per step

the array for remote processes. The local address of "ircbufp" and the remote addresses are stored in the library. Note that since the library is written in C, it is necessary to convert the stored data to a Fortran format array. In line 10, the function "rdma_get_raddr" returns the remote addresses in the C language format. And, in line 11, the intrinsic subroutine "c_f_pointer" converts the C format pointer (ircbufp_cptr) to the Fortran format array (ircbufp_raddr). In line 12, the subroutine "rdma_put_post" transfers data to a specified process and also transfers post information to convey that the communication has ended. In line 13, the subroutine "rdma_wait" waits until the post information is received from a specified process. The communication performed in lines 12~13 is basically the same as the MPI functions in lines 15~17.

In this implementation, 36 MPI function pairs in MODYLAS are replaced with our subroutines and function.

3 EVALUATION

Fig. 3 shows a comparison of the MPI and RDMA communication times in MODYLAS per step using the "water_nve" small data set with three FMM levels on the K computer. The reason for using the "water_nve" is that the effect of communication latency in large-scale execution is examined using a small number of CPUs since the calculation part of MODYLAS is almost scaled to the number of CPUs. In this experiment, one process is assigned to one calculation node, each process has eight threads, and the measurements are performed under strong scaling. From these results, it can be seen that the RDMA communication time is 29~42% less than the MPI communication time. Most communication data sizes are less than 32K bytes, which is a sufficient size to demonstrate the superiority of RDMA in Fig. 1.

Table 1: Calculation time per step (microsecond)

Num. of Proc.	8	16	32	64	128	256
MPI	16,129	9,973	6,941	5,636	4,624	4,151
RDMA (Put)	15,551	9,684	6,706	5,384	4,467	4,033
Improvement	3.72%	2.99%	3.50%	4.68%	3.51%	2.91%

Next, Table 1 shows the total calculation time including the communication time per step. From these results, it can be seen that the RDMA implementation has a higher efficiency than the MPI implementation. Although the efficiency has increased by a factor of 2.91~4.68% overall, this will further increase for calculations with strong scaling with tuned code for hotspot calculations in the future.

4 SUMMARY AND FUTURE WORK

To improve the performance of MODYLAS with strong scaling, we replace MPI communication with RDMA communication provided by the K computer. As a result, we show that the strong scalability for parallel computations with MODYLAS is improved.

The current implementation assumes the use of the extended RDMA interface, so the benefits are obtained only with Fujitsu machines, which use the interface. In order to make MODYLAS available for reducing communication times in various computing environments, we will utilize coarray features of the Fortran standard since the coarray features provide users with one-sided communication, and its implementation may use RDMA that each machine has.

ACKNOWLEDGMENTS

This research was supported by MEXT as "Priority Issue on Post-K computer" (Development of new fundamental technologies for high-efficiency energy creation, conversion/storage and use) using computational resources of the K computer provided by the RIKEN R-CCS through the HPCI System Research project (Project ID: hp180209, hp190174). And, this research was also supported by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" in Japan (Project ID: jh180050-NAJ).

REFERENCES

[1] Andoh Yoshimichi et al. 2013. MODYLAS: A Highly Parallelized General-Purpose Molecular Dynamics Simulation Program for Large-Scale Systems with Long-Range Forces Calculated by Fast Multipole Method (FMM) and Highly Scalable Fine-Grained New Parallel Processing Algorithms. *J.Chem. Theory Compt.* 9, 7 (2013), 3201–3209. https://doi.org/10.1021/ct400203a

[2] Mitsuo Yokokawa et al. 2011. The K Computer: Japanese Next-generation Supercomputer Development Project. In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design (ISLPED '11)*. 371–372.

[3] Naoyuki Shida et al. 2012. MPI Library and Low-Level Communication on the K computer. *FUJITSU Scientific & Technical Journal* 48, 3 (July 2012), 324–330.