Performance Analysis and Portability issues of 64-bit Message Passing Interface for Tera Scale System

PRABU D, ANSHU GARG, VANAMALA V, SRIDHARAN R,
PRAHLADA RAO BB, MOHANRAM N
Systems Software Development Group,
Centre for Development of Advanced Computing (C-DAC), Bangalore, INDIA
'C-DAC Knowledge Park'
No 1, Old Madras Road, Byappanahalli, Bangalore
INDIA

Abstract--- This paper describes the issues underlying the development of 64-bit C-MPI (C-DAC Message Passing Interface) and the porting semantic and strategies of 64-bit C-MPI. This work discusses various issues arising during the porting from 32-bit C-MPI to 64-bit C-MPI that will serve as guidelines to scientific community for their advanced MPI porting work. The ported 64-bit C-MPI is tested, results of some initial experiments comparing 32-bit and 64-bit C-MPI performance under Ethernet and PARAMNet-II interconnect network is presented.

Key-Words--: 64-bit C-MPI, PARAMNet-II, 64-bit MPI, MPI porting, PARAM Padma, C-MPI.

1 Introduction

In the present scenario the MPI applications are emerging with 64-bit compatibility like Climate modeling, Computational fluid dynamics, Nuclear weapon testing etc. There is more demand on 64-bit MPI rather than 32-bit MPI. The new version of 64-bit MPI will provide the way to overcome the 32-bit MPI drawbacks in processing much larger size data. This is accomplished with 64-bit computing machine. Once such machines or nodes are ready, all the underlying software must be converted to 64-bit compatibility. If the 64-bit Supercomputing application has to run on 64-bit CPU, the parallel programming libraries like (Message Passing Interface) and its supporting files and data have to be made compatible to 64-bit operating environment. This enables applications to access and manipulate data items that are upto 64-bit size. The prime advantages of 64-bit MPI porting are given below.

(1) MPI applications will be able to perform computations having larger and more precise data types.

(2) MPI applications will be able to address more Memory, upto 18 Exabytes in 64-bit environment.

(3) Many 64-bit MPI applications will be able to read and write files that are of larger size.

The rest of this paper is organized as follows: Section 2 gives a brief overview of the C-MPI [6]. Section 3 presents the different methods of dealing with the porting task, and the various ways of maintaining backward compatibility with 32-bit C-MPI. Section 4 summarizes the porting efforts and presents in detail the most relevant modifications that had to be made to the C-MPI source code. Section 5 discusses about the porting testbed and performance evaluation of MPI. Section 6 discusses critical porting issues of MPI. The conclusions and future work are presented in Section 7.

2 Overview of C-DAC Message Passing Interface

C-MPI, the message-passing model of parallel computation has emerged as an efficient and recognized paradigm for parallel programming. The Message Passing Interface (MPI) is a standard for message passing, defined by a panel of parallel programming industry leaders including representatives from the national laboratories, universities and key parallel system vendors. Several

parallel applications have been implemented using MPI calls. C-MPI is a high performance implementation of the MPI standard for a Cluster of Multi Processors (CLUMPS). By adhering to the standards, C-MPI supports the execution of the multitude of MPI applications with enhanced performance on CLUMPS. It can run on both Gigabit Ethernet and PARAMNet-II [3].

2.1 MPI Optimization Model

C-MPI optimizes a subset of MPI collective calls by using efficient algorithms for CLUMPS architecture. It also leverages on the fact that most of the high performance networks provide a substantial exchange communication bandwidths. This allows the tuned algorithms to simultaneously send and receive messages over the network, which in turn helps to reduce the number of communication hops. In addition, the algorithms effectively use the higher shared memory communication bandwidths on multi processor nodes. For optimal performance on the PARAM Padma [1] PARAM Padma Supercomputer is a heterogeneous environment having compute nodes running AIX and File Servers based on Solaris with high bandwidth, low latency PARAMNet switch as system area network with MPI software as C-MPI.

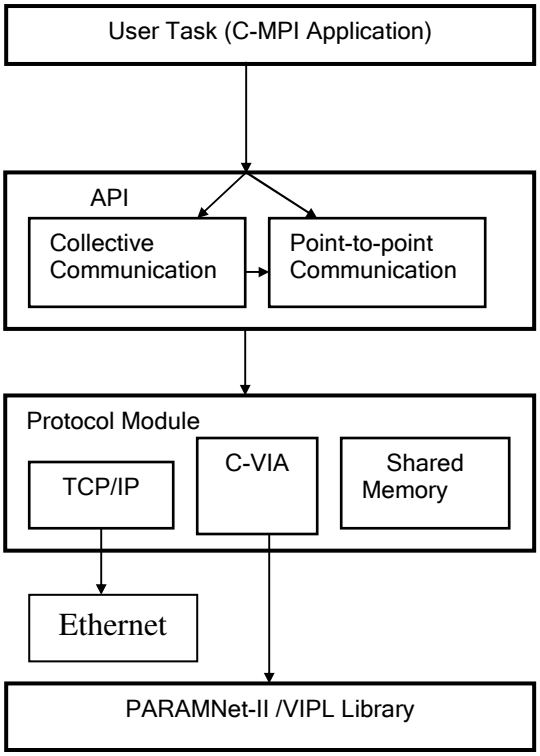

Fig. 1 C-MPI Architecture and Control flow

C-MPI can be operated directly over the high performance system area networks in user space, using lightweight communication protocols. These protocols substantially improve the point-to-point communication performance. As a result, the C-MPI collective calls perform better as compared to the implementation over the TCP/IP. C-MPI architecture as shown in Figure 1 is designed to achieve high performance and portability. It is layered over Abstract Device Interface (ADI) [5] to maintain portability. On C-DAC's PARAM Padma [2], C-MPI employs both TCP/IP [7] and C-VIA in the underlying ADI layer.

The C-MPI functions are implemented in terms of macros and functions. The upper layer does the communication of control information and the lower layer performs the transfer of data from one process address space to another.

2.2 Compatibility With MPI

C-MPI is based on the MPICH [4] implementation from Argonne National Lab [5] and Mississippi State University and implements all MPI functions. The implementation closely follows the published standard and allows programmers to write portable parallel applications, which operate on the PARAM Open Frame system and other Cluster systems supporting the standard. MPI applications only need to be relinked with C-MPI libraries to function on the CLUMPS.

3 Methodology For C-MPI Porting Task

The 64-bit MPI APIs remain virtually identical to the 32-bit MPI APIs. Only a few data types and structures have been changed, to reflect certain items that grow to 64-bit data size. So for most applications, porting boils down to recompiling the source code with applicable compiler options, making sure we have got all the source code we need (or 64-bit binaries for third-party libraries and DLLs you use).

Then compile the source code by enabling –q64 and –qwarn64 switches of the compiler. The –q64 instruct the compiler to recompile source code for the 64bit architecture whereas –qwarn will emit warning about non-portable section of the source code by going through this warning, user can easily identify the portion of the source code and fix it for the 64bit architecture.

4 Porting Efforts

In this section we present the procedure that can be used during efforts to port 32-bit MPI to 64-bit MPI. It is intended as a guideline for projects that deal with porting a similarly large library to 64-bit MPI: Modify the source code lines reported by the Porting procedure discussed in section 3. Most of these changes have to do with replacing the data structures used for storing addresses, replacing network functions for data access, address transformation, and replacing the constants that have been changed for 64-bit MPI.

(1) Make any other necessary modifications in more subtle places not reported by the porting methodology.

(2) Test and debug the code, correcting any issues that arise.

(3) Verify completeness of porting effort.

The first step is to thoroughly read and understand the source code in order to become familiar with the overall structure and techniques used, especially if the person who undertakes the porting task has not been involved in the initial development of the project. The special characteristics of the specific project will determine the most appropriate approach to the porting task. For the initial phases of the porting a parses the source code and reports the source code lines that contain 32-bit-dependent code can prove very useful. The relevant changes are probably rather straightforward, and can proceed in a mechanistic way.

4.1 Modification in the Source Code

We have ported nearly 80% of our existing 32-bit CMPI to 64-bit mode in order to enable it to work in 64-bit environment. Issues faced during this work are as follows.

(1) We have faced segmentation fault while running sample MPI application because of address out of bound. To overcome this changes are made the in header file, mpi-internal.h. The initial structure variable is gprocs[1] to gprocs[1024][mpip_group structure].

(2) C/Fortran interfacing should be Taken care appropriately. Especially pointers from MPI's C code when assigned to pointers in MPI FORTRAN code, vice versa. This is because size of pointer becomes 8-bytes in 64-bit C code, whereas it remains 4-bytes in 64- bit FORTRAN.

(3) pthread_init() problem because of gshmalloc.c. It was returning fixed address, which was causing problem. Problem of zero byte allocation by malloc() has been fixed by making it returning value one byte.

(4) We have rearranged two structures, viz, freelist_t and struct sbpoolcb in mpi_shm.h. Order of structure members is descending.

(5) Using size of () operator instead of Using actual numerical size of variables.

(6) Re-arranging the members of the structures so as to minimize padding.

(7) Limitations with arrays are checked thoroughly.

(8) Pointer and long warnings using '-qwarn64' option should be eliminated.

5 Experimental Setup

The effectiveness of the proposed high performance 64-bit CMPI is tested with the C-DAC's Tera-Scale Supercomputing Facility (CTSF)[2] is located at C-DAC Bangalore, India as shown in Figure 2.

Fig. 2 Picture of C-DAC's Tera-Scale Supercomputing Facility (CTSF)

5.1 Test bed Environment for C-MPI Porting work

PARAM Padma as shown in Figure 3 is C-DAC's High performance scalable computing cluster,

currently operating with a peak computing power of One Teraflop.

Table 1
Description Of PARAM Padma

Specification	Compute nodes	File servers
Configuration	62 nos. of 4 ways SMP and one node. of 32 way SMP	6 nos of 4 ways SMP
No. of processors	248(Power 4@1GHz)	24(UltraSparc-IV@900MHz)
Aggregate memory	0.5 Terabytes	96 Gigabytes
Internal storage	4.5 Terabytes	0.4 Terabytes
Operating system	AIX/LINUX	Solaris
Peak computing power	992 GF (~1 TF)	--

The Computing and storage configuration is shown of PARAM Padma are given in Table 1. Effectiveness of the proposed RSI has been tested with PARAM Padma AIX cluster of 4 ways SMP nodes connected by PARAMNet network as given in Table 1.

5.2 Performance Evaluation and Discussion

The performance indices of 64bit MPI are measured in terms of latency and bandwidth. In this experiment we have tested our 64bit MPI with tested with Latency and bandwidth benchmark on two network interconnects viz., PARAMNet-II and Gigabit Ethernet. A set of test suites has been used to model the performance of MPI point-to-point on PARAM Padma. These suites compare the performance of point-to-point communications, including send and receive overheads for different (contiguous) message lengths and also estimate the network latency and bandwidth. A popular method for measuring communication overheads for a point-to-point communication (e.g., between processor 0 and processor 1) is the round trip scheme. It is observed that the latency (zero byte length latency) at the MPI layer is 25 μs and bandwidth is 80 *MB/sec* using PARAMNet-II network. The overhead measurement time for a MPI blocking point-to-point communication using various message sizes ranging from 0 bytes to 10 MB between two processors of two different nodes of PARAM Padma.

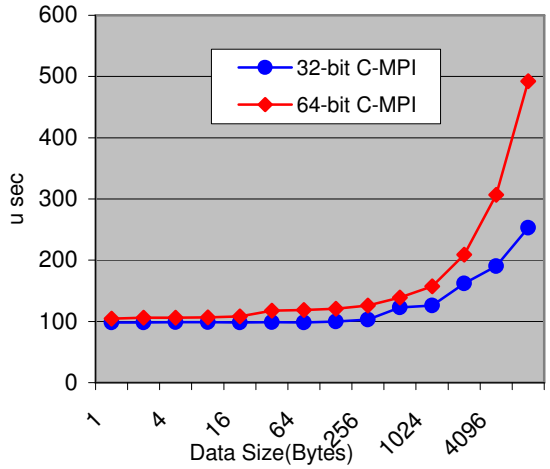

Fig.3 Latency readings of 32-bit and 64-bit C-MPI over Gigabit Ethernet

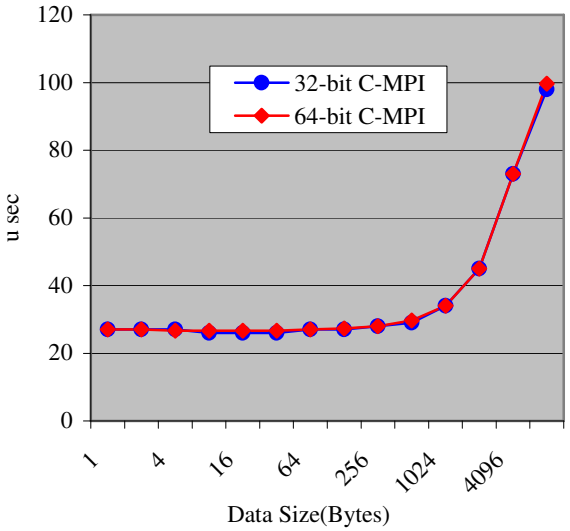

Fig. 4 Latency readings of 32-bit and 64-bit C-MPI over PARAMNet-II

are shown in Fig. 3 and Figure.4 respectively and bandwidth for a MPI blocking point-to-point communication using various message sizes ranging from 0 bytes to 10 MB between two processors of two different nodes of PARAM Padma is shown in figure 5 and 6.

It is found that the bandwidth is good in the case of 64bit CMPI over PARAMNet-II for different

(contiguous) message lengths when compare to 32bit CMPI.

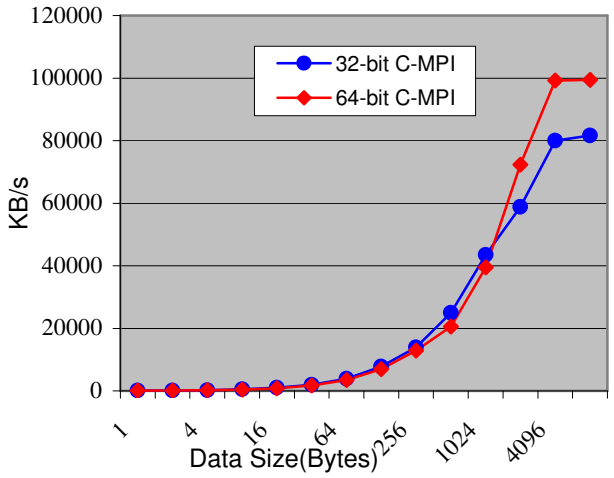

Fig.5 Bandwidth reading of 32-bit and 64-bit C-MPI overGigabit Ethernet

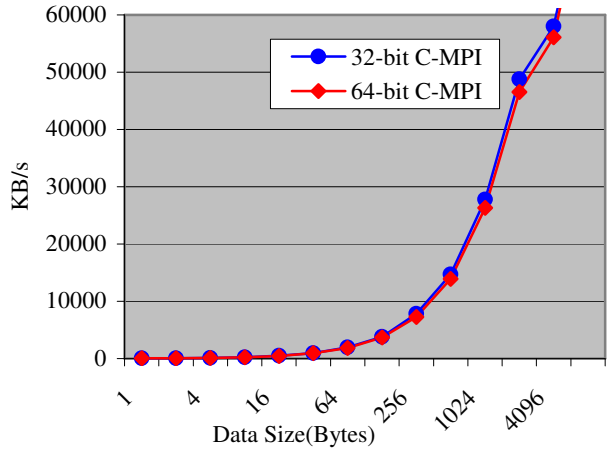

Fig.6 Bandwidth readings of 32-bit and 64-bit over PARAMNet-II

6 Porting Issues

There are two kinds of issues that introduce difficulties for the porting effort: isolating the Structures and function that have to be modified, and the fact that some of the indirect dependencies might be scattered or affect large portions of the source code. In this section we report some of the most difficult issues that were revealed during the porting of the C-MPI project that required changes not easily identifiable from the beginning.

(1) In C-MPI the sending process completes and looping in MPI Finalize. Receiving process looping in MPIRecv. This is because the numbers of descriptors available at receiving end are not enough. After a great struggle we have identified and changed DS[2] to DS[4] in /usr/include/vipl.h

(2) We encountered segmentation fault during NAS benchmarking in our PARAM Padma Cluster using C-MPI over Ethernet network. The NAS Parallel Benchmarks suite is widely used to compare the performance of all types of parallel computing platforms, since it contains computational kernels that are representative of several different algorithms used in real-world applications. NAS 2.4 constitutes eight CFD problems, coded in MPI and Standard Fortran77/C. We rectified it changing mpi-internal.h, mpi-fortan.h, mpi.h, fhandle.c, request.c ,waitall.c , waitall_f.c and isend.f.c

(3) While benchmarking with P-COMS v-1.1.3 (**P**ARAM - **C**ommunication **O**verhead **M**easurement **S**uites) – a set of test suites are been used to model the performance of MPI point-to-point and collective communications on PARAM Padma. These suites compare the performance of point-to-point communications, including send and receive overheads for different send and receive modes and for different (contiguous) message lengths and also estimate the network latency and bandwidth. Many times we faced problem like programs exiting without giving error message. After a long struggle we identified this problem and changed, an If condition of our code from "size<PI_BSEND_OVERHEAD" while testing oneway_buff_send in buffersend

(4) While testing C-MPI over PARAMNet using NAS Benchmarking on PARAM Padma, we faced problem with parament code error=0x101(remote descriptor error). Changed to MAX_DATA_SIZE= (1024*16)- sizeof (paramnet descriptor) – sizeof (paramnetpacket_eager_start) from MAX-DATA_SIZE = ((1024*16))-64 sizeof(viadev_packet_eager_start) in mpi/engine/cvia/defines.h. Here 64 is sizeof (paramnet descriptor) in bytes for 32-bit and 88-bytes in 64-bit.

There were also a few dependencies of this kind scattered in a lot of other classes, mainly in the library. Because such dependencies are not detectable

using an automated tool and because of the very large size of the source code base, this was the most time-consuming part of the modifications in the source ode and also the cause for a number of bugs that appeared during the porting.

7 Conclusions And Future Work

In the present scenario it is inevitable for all scientific applications to work under 64-bit environment for it large addressable memory, Computational speed, large file size We believe that our MPI porting strategies will be useful and may provide rule of thumb for the scientific community for migrating the 32-bit MPI to 64-bit MPI. In Future work, we have planned for Grid enabled 64-bit C-MPI [10] for the Supercomputing community.

References:

[1] PARAM Padma Supercomputing Cluster, (C-DAC) India, http://www.cdac.in/html/parampma.asp

[2] C-DAC Terascale supercomputing facility CTSF, www.cdac.in/html/ctsf/resource.asp

[3] High Speed Network ,PARAMNet-II, www.cdac.in/HTmL/pdf/PARAMNet.pdf

[4] W. Gropp, E. Lusk, N. Doss, A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard",in Parallel Computing, vol. 22, no. 6, pp. 789-828, September 1996

[5] William Gropp, Ewing Lusk “MPICH Abstract Device Interface, Version 3.3,” MCSD, Argonne National Laboratory, December 2001. http://www.cse.ohio-state.edu/~panda/788/paper/c_adi3man.pdf

[6] C-DAC Message Passing Interface, http://www.cdac.in/html/ssdgblr/cmpi.asp

[7] TCP/IP. Available at http://www.ietf.org/rfc/rfc1180.txt

[8] NAS, Parallel Benchmark www.nas.nasa.gov/Software/NPB

[9] PARAM- Communication Overhead Measurement Suites (P-COMS), Centre for Development of Advanced Computing, Pune, India. http://www.cdac.in/html/betatest/hpc.asp

[10] ‘Garuda’, The National Grid Computing Initiative, CDAC Bangalore INDIA http://www.garudaindia.in/tech research.asp