CGUW: A System Software for Heterogeneous IPC Mechanism in Grid Computing Environments

Ehsan Mousavi Khaneghah
Department of Computer Engineering
Shahed University
Tehran, Iran
emousavi@shahed.ac.ir

Seyed Ali Ghoreishi
Department of Computer Engineering
Shahed University
Tehran, Iran
s.a.ghoreishi1995@gmail.com

***Abstract*—Interprocess communication mechanisms, one of the effective factors for response time in High performance computing. Existing standards for interprocess communication between a heterogeneous operating systems such as MPI implementation at library level lead to lower performance and Increase response time compared to implementation at the operating system kernel level. We propose an approach to enable the use of IPC in heterogeneous distributed systems such as grid computing. This mechanism uses distributed shared memory for communication between Linux and Windows operating systems. Windows IPC mechanism uses shared memory and remote procedural call. Linux Inter-process communication (IPC) converting Windows remote procedure call (RPC) to Proportional System V shared memory in kernel level. We propose wrapper for the use of our mechanism with distributed application.**

Keywords— Interprocess communication; Distributed Shared Memory; Wrapper; Kernel Level

I. INTRODUCTION

High Performance Computing (HPC) use parallel computing to run complex computation scientific problems. High Performance Computing decomposes the problem into several subproblems that run on various computational units to complete in an acceptable time [1]. In HPC, computational units need to collaborate and communicate frequently to accomplish a specific task, for this reason, IPC is the main part of HPC Manager [2]. Performance, response time and efficient implementation of IPC is vital for HPC [3], [4]. IPC has been implemented at different levels namely at the user level as a set of library routines, at the operating system interface level as a transparent middleware, or at the operating system kernel level transparent to users and applications [5]. User level implementation of distributed IPC is the least efficient and easiest to do, though hardest to be used by programmers.

Kernel level implementation is the most effective and easiest to be used by programmers, but it has the hardest implementation. Middleware level implementations are somewhat in between these two implementation levels considering their efficiencies and their ease of use [5].

Distributed HPC such as Grid and P2P, utilizes two main programming paradigms, which are the distributed shared memory and Message passing paradigms [1]. Distributed shared memory abstraction is different from message passing, but underlying utilization of system resources is similar [2]. In Distributed shared memory, processes collaborate to each other by sharing the same memory segment. Data does not need to be copied from one process memory space to another [1].

On the other hand, a message passing communicating processes share data through sending messages. Message passing provides transparent communication interface. Underlying message passing implementation managed name resolution and shared data between processes [2]. Message Passing Interface (MPI) is the de-facto standard for application development in high performance clusters [6]. MPI provides point-to-point, collective, and one-sided types of communications. Point-to-point operations support both blocking and non-blocking modes [1]. Grid computing has focused on data sharing, collaboration among distributed organizations in Different geographical places [7], [8].

A Grid is a large-scale resource sharing and problem-solving mechanism in virtual organizations [7]. Grids can be considered to be of two types, Computational and Data Grids. A large number of computers are linked globally to form a computational Grid. Software Distributed Shared Memory systems provide a compromise between message passing hardware simplicity on multicomputer and shared memory programmability on multiprocessors [7], [9].

Many software DSM systems, such as Ivy [10], Midway [11], TreadMarks [12], Munin [13], and JIAJIA [14] have been implemented on the top of message passing. Software distributed shared memory allow shared memory programs to run on non-shared memory clusters. Using benefits of the performance of message passing hardware and shared memory conceptual model. These software Limited scalabilities and lack of standard API [7], [15].

For grids to become commonly accepted there is a need for implementations of well known, easy-to-use programming environments, such as a (virtual) shared memory environment [7]. Some scientific research is requiring special software to specific calculations such as for simulation, rendering. This software requires HPC resource and specific operating system for running. Clusters choice for institutions to provide High Performance Computing resources.

The Scientific application needs to be run on only open source operating systems such as Linux or only commercial operating systems such as Windows or multi-platform

978-1-5386-0774-9/17/$31.00 ©2017 IEEE

operating systems. Some closed-source non-Java large-scale Windows software such as Backburner [16] for 3ds Max [17] to rendering in distributed network, only run on Windows platform. Some software such as Matlab to analyze and design the systems supported multi-platform [18].

The first solution to solve these problems is the virtualized system implementation [18]. Virtualization cannot run effectively on legacy hardware. Many legacy Beowulf clusters do not have the latest hardware to run multiple operating systems on one machine [18]. In the virtual machine, each virtual OS executes the instructions, transferring the signal to the hardware, but there only one real hardware exists, if too many virtual machines run on the same system, the I/O operation, the resources of CPU, even the volume of RAM all become the bottleneck.

The virtualization technologies bring out the performance deterioration in clusters [19]. Because of performance overheads, VM technologies useless in HPC environments [20]. The emulator has to intercept most of the instructions used for guest-kernel to application communications and has to simulate a complete set of virtual hardware devices; this emulation implies several layers are imposing a natural lower bound for extra network latency, which cannot be undercut.

Another solution is creating sub-clusters for each platform to running packages on different operating systems. This solution has weak utilization of the resources [18]. Having efficient run times and transparently communication data between nodes is important in heterogeneous clusters [6].

The rest of the paper is organized as follows: In Section II reviews some previous works about inter-process communication on a cluster or Grid and Expresses advantages and disadvantages of these mechanisms; In Section III introduces our proposed wrapper structure and each of its modules operation. In Section IV, Discussion why we chose this mechanism; In Section V, summarizes our proposed mechanism features, results, and future work.

II. RELATED WORK

Some work provides a transparent mechanism for inter-sites process communication. These works perform cross-site MPI execution over different cluster sites. MPICH-G2, MPIg, and MUSCLE-HPC use the Globus Toolkit services to perform cross-site executions. They enable the user to run parallel code across multiple and heterogeneous clusters sites using the same command as in a parallel machine. In some HPC machines like Cray, MPI.2-2 features are not implemented in the installed MPI programming environment [21].

In [19] focuses on building an MPI based heterogeneous cluster with the MPICH2. The developers of MPICH2 provide a vital hybrid process management called SMPD, which can be used in UNIX, Linux, and Windows. For implementation these mechanisms, MPI standard library must be installed on every machine in the cluster. These mechanisms use library level inter-process communication that less efficient than kernel level IPC. These mechanisms realm of clusters, not for Grid computing.

In [22] presents VMRPC, a lightweight RPC framework circumvents redundant data copy and serialization/deserialization by leveraging the heap and stack-sharing mechanism [23]. Their evaluation shows that the performance of VMRPC is an order of magnitude better than traditional RPC systems. VMRPC trades transparency for efficient communications and provides fast responsiveness and high throughput when deployed for communicating components in virtualized environments. It is specifically designed for the applications that require high-volume data transfer between VMs.

Hybrids Operating System in Grid environment runs well without virtualization because virtualization has performance overheads. VM cannot run effectively in Grid environment because many machines in Grid have legacy hardware and little have the latest hardware.

In [18] presents dualboot-oscar, dual-boot cluster middleware for multi-platform systems in HPC computing. Reboot takes time 5 minutes, and we do not use the resource of the machine in rebooting time on the cluster. This mechanism uses the fat partition for creating partitioning for Linux, Windows and shared partition for every machine in the cluster and only suitable for the use of legacy hardware.

In [5] developed MS-Windows Wrapper Manager (WWM) and the Linux Wrapper Manager (LWM) for converting MS-Windows IPC call into equivalent Linux IPC calls and benched their approach on a hybrid computer cluster running both types of operating systems (Windows and Linux). Use RPC for developing programs on MS-Windows and message mechanism for Linux-based programs. Divide cluster into smaller sub cluster for each platform; lead to decreased utilization of the cluster resources. Message passing does not match mental model of the user for programming. Heterogeneous IPC mechanism more required in Gird computing than cluster computing.

In [24] presents SMG (Shared Memory for Grids), present Open MP compatible DSM system layered use of standard message passing interface library which can execute in a heterogeneous grid environment. Because of use of MPI library, this DSM system not efficient enough than kernel level DSM implementation. This MPI implementation does not support multithreaded applications.

In [25] presents Teamster-G, DSM system that allows DSM programs run on the virtual dedicated homogeneous grid and does not support Heterogeneous Grid.

In [7] presents SMIG (Shared Memory Integrated with Grid). SMIG application run in the background with low priority in Grid machine. SMIG model easy install and easy to use. This model designs only for homogeneous LAN environment in grid computing and does not support heterogeneous WAN environment in grid computing. This model is run on only Linux platform.

In Table I, the comparison between related works in terms of Suitable for Grid Computing, run in a heterogeneous environment, use the standard library (such as MPI) and supported DSM is shown.

TABLE I. COMPARISON BETWEEN SOME IPC MECHANISMS

Mechanism	Grid Computing	heterogeneous environment	MPI	DSM
MPICH-G2, MPIg, MUSCLE-HPC	No	Yes	Yes	No
VMRPC	No	Yes	No	No
dualboot-oscar	No	Yes	No	No
WWM and LWM	No	Yes	No	No
SMG	Yes	Yes	Yes	Yes
Teamster-G, SMIG	Yes	No	No	Yes

As shown in Table I, some mechanisms run in Grid Computing use DSM, the other mechanisms run in heterogeneous cluster computing with IPC library implementation overhead. VMRPC and dualboot-oscar have System resources and time overhead. WWM and LWM run in heterogeneous cluster rarely used by distributed applications.

III. THE PROPOSED WRAPPER

We propose a CGUW (cluster grids universal wrapper) mechanism for efficient communications between processes that run on distributed HPC such as Grid computing environments that contain Windows operating system and Linux operating system. This mechanism implemented distributed shared memory in a heterogeneous grid environment. Scientist designs their program with CGUW library and our programming model.

We use DSM to provide easy-to-use programming environments that became commonly accepted between many sciences in grid computing. The process that is running on Linux and Windows can transparently communicate and user easy developing their application in grid environment by using CGUW. Distributed IPC mechanism is implemented at the kernel level of Linux can provide high performance computing for our mechanism [4].

A. Primitive mapping (RPC and DSM in MS Win and DSM in Linux)

RPC is a mechanism for IPC in windows and to be used by the programming Language. We want map RPC on Windows OS to shared memory on the Linux OS. RPC system calls that have equivalent shared memory system calls easy map together but if system calls on one side do not have any equivalent or more than one Equivalent on another side, the mapping is complicated. In Table II, RPC main functions operation on MS Win operating system are shown.

TABLE II. RPC FUNCTIONS IN WINDOW [26]

Function	Explanation
RpcBindingFromStringBinding	Gets string binding handle And returns server binding handle [27]
RpcBindingFree	releases memory used by a server binding handle
RpcBindingToStringBinding	Gets server or client binding handle and Returns a string binding handle [28]
RpcStringBindingCompose	Gets network options and returns string binding handle [29]

As shown in Table II, the RpcBindingFromStringBinding function creates a server-binding handle. RpcBindingFree function releases the handle. RpcBindingToStringBinding returns a string-binding handle. RpcStringBindingCompose function combines the components of the binding handle into a string, and RpcNsBindingImportNext function looks up an interface and returns a binding handle.

Shared memory main functions operation on Unix System V operating system is shown in Table III.

TABLE III. SYSTEM V SHARED MEMORY SYSTEM CALL IN LINUX [30]

Function	Explanation
shmget	creates a new shared memory segment or obtain the identifier of an existing segment
shmat	Attaches segment to virtual memory of related process [30]
shmdt	detaches the shared memory segment
shmctl	deletes the shared memory segment

As shown in Table III, the shmget function creates or opens shared memory segment, shmat function attaches segment to process address space, shmdt function detaches segment from the process, and shmctl function deletes segment.

In Table IV, shared memory main functions operation on MS-Windows operating system is shown.

TABLE IV. SHARED MEMORY SYSTEM CALL IN WINDOWS [31]

Function	Explanation
CreateFileMapping	Returns a handle to opened or Created file mapping object [32]
MapViewOfFile	Maps a view of a file mapping into the address space of a calling process
UnmapViewOfFile	Unmaps view of file mapping from address space of calling process [33]
CloseHandle	Closes an open object handle

As shown in Table IV, CreateFileMapping function Creates or opens file mapping object, MapViewOfFile function Maps a view of a file into process address space, UnmapViewOfFile function unmaps a view of a file, and CloseHandle function closes an object handle.

B. CGUW Structure

Microsoft Converter and Executor are two main modules in Wrapper structure. The Converter module takes RPC from RPC server and converts it to System V shared memory or vice versa. The Executor module runs on Linux kernel to execute windows client application. Windows client has Windows Server ID manager to find a server for the client request. Fig. 1 shows our wrapper mechanism and completely operating scheme at a glance.

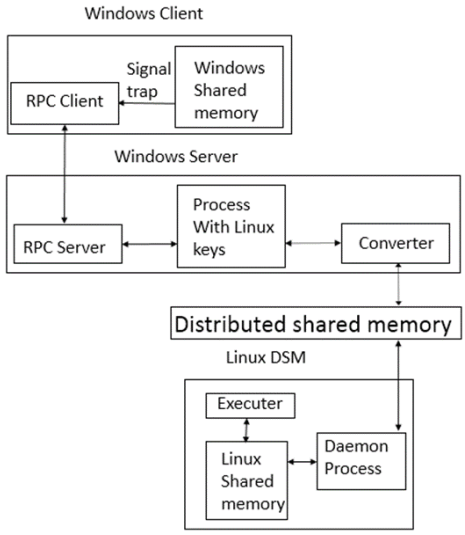

Fig. 1. CGUW operating scheme

As shown in Fig. 1, CGUW has three part. Windows client Windows Server and Linux DSM. In Windows client, user distributed program use windows shared memory. If the name of the file-mapping object to be opened does not exist, our wrapper catches signal trap and sent it to RPC client process. RPC mechanism building block of the Windows communication structure.

RPC establishes a connection between Windows client and suitable windows server by Windows Server ID manager in a grid environment and sends the name of file mapping requested by distributed program in windows client to windows server. In windows server, RPC Server sends object name to process with Linux Key, and this process finds suitable Linux to DSM communication with it.

Converter module takes RPC server system call and converts it to proportional System V shared memory system call. In Linux DSM, Daemon process is running and waiting for a connection from windows server. When the connection is established, Daemon process calls appropriate file mapping object. The executor is running user program in kernel level efficiently. Distributed program can run on Windows and Linux and communicate with DSM in grid environment with our proposed Wrapper.

IV. Discussion

Our proposed Wrapper uses RPC and DSM to create Grid based programs running on a heterogeneous Grid. DSM share keys between processes when it was created. We need two servers for Maintenance List of DSM keys made by Windows and Linux machines. These Servers should have addresses of each other. This approach use centralizes mechanism.

For this reason, it loses its effectiveness when scalability occurred. Servers with a list of DSM keys are a bottleneck for support openness in our proposed system. Programs on Grid computing have these properties: (1) Correspond to the conceptual model of a programmer for programming distributed programs (2) use effective resource usage model (3) Provide efficient facilities for establishing communication between processes in the Grid environment. Our wrapper has these properties. Scientific Windows-based programs can use features and resources of Linux server by using our wrapper in Grid computing.

V. Conclusion and Future Work

Distributed programs run on heterogeneous Grids have more flexible and various facilities than homogeneous Grids. To ease of higher performance, we use Kernel level implementations Instead of user level implantation and to provide ease of programming; we use DSM and wrapper. The wrapper is establishing communication between the closed source and open source operating systems in Grid computing that supported programming on heterogeneous distributed systems. DSM makes inter-process communications transparent to end-users; we selected DSM as a suitable communication mechanism in our wrapper solution.

Our mechanism consists of RPC for developing programs on MS-Windows and DSM for Linux-based programs. RPC calls from Windows client to the server are converted into equivalent DSM calls for Linux server using a mapping table.

In our future works, we intended to use our approaches on the internet of thing. Android devices have slow processing speed and memory than Linux servers to scientific calculation program for things to make smart decisions in IOT. Client Android-based devices can communicate to Linux servers for effective Execute task and use facilities of Linux servers.

References

[1] L. Alawneh, A. Hamou-Lhadj , J. Hassine, "Segmenting large traces of inter-process communication with a focus on high performance computing systems," *Systems and Software,* 2016.

[2] A. Hammar, Analysis and Design of High Performance Inter-core Process Communication for Linux, Uppsala university, 2014.

[3] E. Mousavi Khaneghah, N. Osouli Nezhad, S. Leili Mirtaheri, M. Sharifi and A. Shirpour, "An Efficient Live Process Migration Approach for High Performance Cluster Computing," *International Conference on Innovative Computing Technology,* 2011.

[4] S. L. Mirtaheri, E. Mousavi Khaneghah, M. Sharifi, "A Case for Kernel Level Implementation of Inter Process Communication Mechanisms," *Information and Communication Technologies: From Theory to Applications,* 2008.

[5] M. Sharifi, E. Mousavi Khaneghah, M. Kashyian, S. L. Mirtaheri, "A platform independent distributed IPC mechanism in support of programming heterogeneous distributed systems," *J Supercomput,* 2012.

[6] D. Sankar Banerjee, K. Hamidouche, D. K. Panda, "Designing High Performance Communication Runtime for GPU Managed Memory: Early Experiences," *General Purpose Processing using Graphics Processing Unit Processing using Graphics Processing Unit,* 2016.

[7] S. Shah, S. Mahajan, "Resource Optimization in a LAN Environment Using SMIG-Shared Memory Integrated with Grid," *International Conference and Workshop on Emerging Trends in Technology,* 2011.

[8] V. Chaudhary, H. Jiang, "Techniques for migrating computations on the grid," *Communications of the ACM,* 2006.

[9] W. Hu, W. Shi, Z. Tang, "JIAJIA: A Software DSM System Based on a New Cache Coherence Protocol," in HPCN Europe '99: Proceedings of the 7th International Conference on High-PerformanceComputing and Networking, 1999.

[10] K. Li, "IVY: A shared virtual memory system for parallel computing," *International Conference on Parallel Processing,* 1988.

[11] B. N. Bershad, M. J. Zekauskas, W. A. Sawdon, "The Midway Distributed Shared Memory System," *Digest of Papers. Compcon Spring,* 1993.

[12] P. Keleher, A. L. Cox, S. Dwarkadas, W. Zwaenepoel, "TreadMarks: distributed shared memory on standard workstations and operating systems," *WTEC'94 Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference,* 1994.

[13] J. B. Carter, "Design of the Munin Distributed Shared Memory System," *Journal of Parallel and Distributed Computing,* 1995.

[14] M. R. Eskicioglu, T. A. Marsland, W. Hu, W. Shi, "Evaluation of the JIAJIA software DSM system on high performance computer architectures," *System Sciences (HICSS), Annual Hawaii International Conference on,* 1999.

[15] L. Kontothanassis, R. Stets, G. Hunt, U. Rencuzogullari, G. Altekar, S. Dwarkadas, M. L. Scott, "Shared memory computing on clusters with symmetric multiprocessors and system area networks," *Shared memory computing on clusters with symmetric multiprocessors and system area networks, ACM Transactions on Computer Systems (TOCS),* 2005.

[16] "Autodesk Backburner," 10 January 2017. [Online]. Available: https://apps.autodesk.com/3DSMAX/en/Detail/Index?id=3481100546473279788&appLang=en&os=Win64.

[17] "Autodesk 3ds Max," 10 January 2017. [Online]. Available: http://www.autodesk.com/products/3ds-max/overview.

[18] S. Liang, V. Holmes, I. Kureshi, "Hybrid Computer Cluster with High Flexibility," *Cluster Computing Workshops,* 2012.

[19] Y. Guo, D. Hu, P. Wu, "MPI-Based Heterogeneous Cluster Construction Technology," *Distributed Computing and Applications to Business, Engineering & Science,* 2012.

[20] W. Huang, M. J. Koop, Q. Gao, D. K. Panda, "Virtual Machine Aware Communication Libraries for High Performance Computing," *ACM/IEEE Conference on Supercomputing,* 2007.

[21] M. Ben Belgacem, B. Chopard, "A new high performance API to couple multiscale parallel applications," *Future Generation Computer Systems,* 2017.

[22] V. Masne, R. Phasate, S. Thombre, N. Sambhe, "RPC System for Distributed Network Systems," *IJIRSET,* 2016.

[23] L. He, K. Li, L. Shi, J. Sun, H. Chen, "A Fast RPC System for Virtual Machines," *IEEE Transactions on Parallel & Distributed Systems,* 2013.

[24] J. Ryan, B. Coghlan, "SMG: Shared Memory for Grids," 2004.

[25] T. Lianga, C. Wub, C. Shiehb, J. Changc, "A grid-enabled software distributed shared memory system on a wide area network," *Future Generation Computer Systems,* 2007.

[26] "RPC Functions," 10 January 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa378623(v=vs.85).aspx.

[27] 4 February 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa375590(v=vs.85).aspx.

[28] 4 February 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa375612(v=vs.85).aspx.

[29] 4 February 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa378481(v=vs.85).aspx.

[30] M. Kerrisk, The Linux Programming Interface, San Francisco: No Starch Press, 2010, pp. 997-1015.

[31] "File Mapping Functions," 10 January 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa366781(v=vs.85).aspx.

[32] "CreateFileMapping function," 4 February 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa366537(v=vs.85).aspx.

[33] "UnmapViewOfFile function," 4 February 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa366882(v=vs.85).aspx.