Memory Management for Memory-based Inter-Task Communication on the Hybrid Operating System Node

Mikiko Sato
Tokyo University of Agriculture and Technology
mikiko@namikilab.tuat.ac.jp

Kazumi Yoshinaga
Yuichi Tsujita
Atsushi Hori
Riken AICS
{kazumi.yoshinaga, yuichi.tsujita, ahori}@riken.jp

Mitaro Namiki
Tokyo University of Agriculture and Technology
namiki@cc.tuat.ac.jp

1. INTRODUCTION

Hybrid architecture system has been designed for highly parallel processing. On such system, multiple OSs manage each CPU and memory resources. Therefore, tasks running on each OS have to communicate specifically between CPUs [1]. To prepare common data space on multiple CPUs, DSM [2] and a single system image (SSI) [3] have been proposed. However, these approaches require the management overhead for the data copy for the guarantee of the consistency on the mutual physical memory. The aim of this study is to provide a program execution environment that enables mutual collaboration between tasks on different OSs with little overhead. This study proposes a page management for the global virtual address space managed by multiple OS on each CPU. And this approach enables memory-based communication between mutual tasks and realizes application programs effectively utilizing both CPUs on the hybrid architecture system.

2. DESIGN

Our approach is that a single global virtual address space structured by more than one virtual address space on the different OS is managed by light-weight method. Each OS manages page tables and establishes the page table entry to access another address space. To maintain the consistency of the page table, only entry of the remote memory accessed by a local task is added to the local page table. And to reduce a memory management overhead, the whole range of the global address space is pinned down. Assignment of physical page and overhead occurrence with page-in are avoided at the time of memory access with this design.

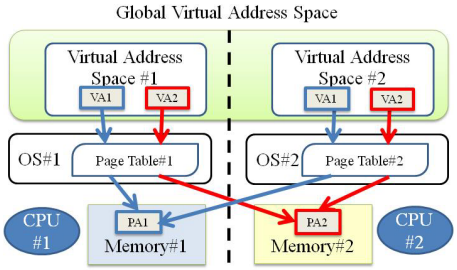

Figure 1. Remote Memory Access with Local Page Table.

3. IMPLEMENTATION AND EVALATION

The page table structure in Multiple PVAS (M-PVAS) [4] which applied this study is shown in Fig. 2. It is implemented to the Linux Kernel on Xeon (X5680, 3.33GHz) and Xeon Phi (5110P, 1.053GHz). In order to investigate the memory management overhead on different CPUs, the processing time of the page fault handler was evaluated in M-PVAS. The execution time of a page fault handler is 1.9us and 8.0us on Xeon and Xeon Phi, respectively. As shown in Table 1, most time was spent on page table reference of remote page table and it was heavy. So it's necessary to reduce the number of the page fault times for remote memory accessing. On the M-PVAS, the tasks on the same CPU can share the same page table, and each OS processes a page fault handling for the remote RAM only the accessed address to eliminate the page fault overhead.

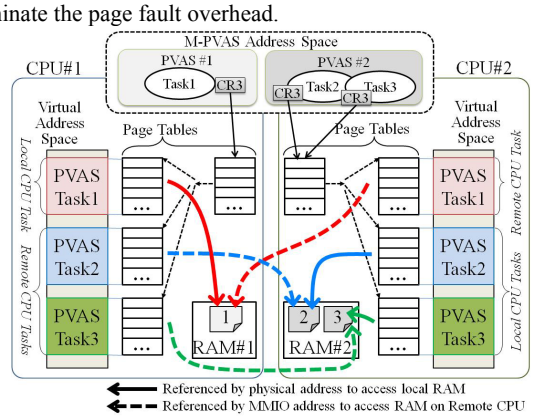

Figure 2. Page Table Structure.

Table 1. The processing time details of the page fault handler

	Xeon	Xeon Phi
Get and release Lock	0.09us	0.7us
Page table walk and check entry(local page table)	0.04us	0.3us
Page table walk and check entry(remote page table)	1.4us	4.9us
Add entry information	0.4us	2.1us
Total	1.9us	8.0us

4. CONCLUSIONS

This paper proposed the global virtual address space managed by multiple OSs for the hybrid OS architecture, and showed the basic evaluation of the memory management overhead. In the future work, the effect of this method will be evaluated using application programs on a hybrid system.

5. ACKNOWLEDGMENTS

This research has been partially supported by JST, CREST (Japan Science and Technology Agency, CREST).

6. REFERENCES

[1] J. Reinders, An Overview of Programming for Intel Xeon processors and Intel Xeon Phi coprocessors, Intel (2012).

[2] S. Yan, et al., Optimizing a shared virtual memory system for a heterogeneous cpu-accelerator platform. SIGOPS Oper. Syst. Rev., 45(1), pp.92-100, Feb. 2011.

[3] ScaleMP. ScaleMP Tech Overview (PDF, complete). http://community.hartree.stfc.ac.uk/access/content/group/admin/HPC Training/ScaleMPtrainingcourse2013/ScaleMP_Tech_Overview.pdf.

[4] M. Sato, et al., Design of Multiple PVAS on InfiniBand Cluster System Consisting of Many-core and Multi-core, EuroMPI/ASIA '14, pp.133-138, Sep. 9-12, 2014.