Deployment and Evaluation of Azalea multi-kernel for manycore

Seung Hyub Jeon, Seung-Jun Cha, Ramneek, Yeon Jeong Jeong, Jin Mee Kim, Sungin Jung
Electronics and Telecommunications Research Institute (ETRI)
Daejeon, South Korea
{shjeon00, seungjunn, ramneek, yjjeong, jinmee, sijung}@etri.re.kr

***Abstract*—In a manycore environment, the monolithic kernel has problems in terms of scalability. To solve this problem, we propose an Azalea multi-kernel. The Azalea multi-kernel takes all the advantages of multi-kernel and unikernel. First, intensive I / O offloads to the full-weight kernel to improve the locality of the kernel service. Second, application performance is enhanced by removing kernel noise. In this paper, we describe the characteristics of the Azalea multi-kernel and demonstrate its feasibility in terms of performance and scalability with existing Linux through the experiment of deploying a full-weight kernel(Linux) and three unikernels(azalea unikernel) in a single node.**

***Index Terms*—manycore, scalability, multi-kernel, unikernel**

I. Introduction

Monolithic kernels such as Linux have shown the problem of scalability due to cache coherency and data sharing as the number of cores grows [1] [2] [3]. In order to solve this problem, research has been conducted on multi-kernels consisting of one or more kernels that provide different functionalities. Scalability can be improved by minimizing data sharing and maximizing the locality of kernel services [5] [6] [7].

Apart from this, there was an effort to minimize the noise of specialized kernels used in multi-kernels, one of which is unikernel [8] [9] [10] [11]. Unikernel is based on the library OS and allows only the services required by the application to be included, minimizing switching between kernel and user.

In this paper, we propose Azalea multi-kernel using full weight-kernel specialized to I/O and unikernel specific to computation for the scalability of a manycore system. And we'll show how to deploy it on a single Knight landing server.

The rest of this paper is structured as follows. In section II, we briefly review backgrounds: multi-kernel and Azalea unikernel. In section III, we present the functionality of azalea multi-kernel. Section IV includes the experimental results and discussion, followed by a conclusion in section V.

II. Background

A. Multi-kernel for manycore

A multi-kernel refers to a kernel consisting of one or more kernels with different functions. This tries to provide scalability by increasing the locality of service. Such multi-kernels include FusedOS [5], McKernel [6], and mOS [7], etc.

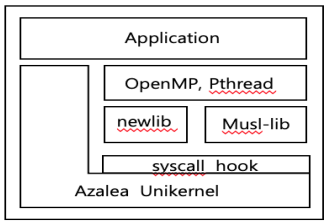

Fig. 1. newlib, musl-lib and syscall hook

B. Azalea unikernel

Azalea unikernel [13] is a library OS that minimizes the copying of data between kernel and user by using a single address space. It also enhances application performance by eliminating mode switching between kernel and users. By doing this, it is lighter than a lightweight kernel. Another goal of Azalea-unikernel provides compatibility with existing Linux applications. To do this, some POSIX system calls are supported.

III. Azalea multi-kernel

Azalea multi-kernel supports the following functions to optimize to run existing applications in a manycore environment.

A. sideloader

The sideloader is a resource manager that allocates CPU and memory to run a unikernel and reclaims resources when the application is terminated. In a single node deployment environment, it is implemented as a module of full-weight kernel(Linux).

B. newlib/musl-lib support

The biggest problem with the newly developed kernel is that it needs to support various libraries like glibc used in existing applications to provide compatibility. To do this, Azalea unikernel supports a simpler and widely used alternative library, newlib [14] and musl-lib [15]. On top of this library, we provide a parallel execution environment using embedded pthread [17] or OpenMP library. A syscall called from newlib / musl-lib is hooked and replaced with a kernel function call. We will replace syscall at runtime with binary rewrite [16] later. Table 1 lists the currently supported system calls.

TABLE I
LIST OF SUPPORTED SYSTEM CALLS

Category	System Calls
Azalea Unikernel Internal (POSIX:11)	brk, clone, exit, getpid, mmap, munmap, getprio, setprio, msleep, kill, signal
Azalea Unikernel Internal (11)	get_ticks, block_current_task, reschedule, wakeup_task, yield, sem_init, sem_destory, sem_wait, sem_post, sem_timewait, sem_cancelablewait
File system Offloading (20)	read, write, open, creat, close, lseek, lstat, mkdir, link, unlink, readlink, openat, uname, getcwd, chdir, access, stat, sync, fstat, ioctl
Network Offloading (8)	gethostname,gethostbyname,getsockname, socket, bind, listen, connect, accept

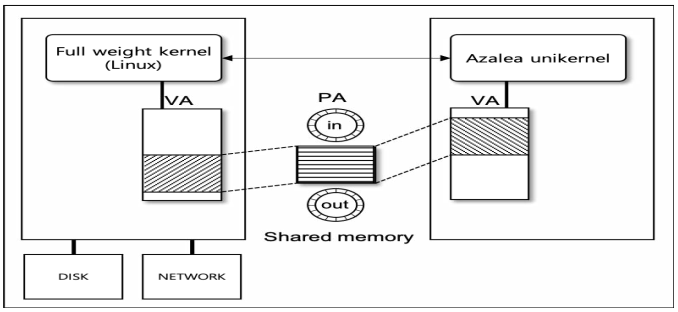

Fig. 2. Performance comparison Between Azalea and Linux

C. I/O offloading

For file I/O or network I/O that is not supported by Azalea unikernel, the system calls are offloaded to the full-weight kernel and returned after processing. As shown in Fig 2, it uses queue-based message passing using shared memory and enhances performance with minimal memory copying to improve I/O performance.

IV. EXPERIMENTAL RESULTS

This section shows the potentiality of Azalea multi-kernel by comparing performance and scalability of AIM Multiuser benchmark (AIM7) application on Xeon Phi server.

A. Hardware

We used an Intel Xeon Phi Server (7290f [18], 72 cores, 1.5GHz, 192GB RAM) as the test environment. When Azalea multi-kernel is applied, it is made up of four quadrants using sub-NUMA cluster mode(SNC-4) in Xeon Phi [19]. We run Linux on one of the four quadrants as full-weight kernel and run Azalea unikernel on each of the remaining three quadrants. The reason for doing this is to maximize the locality.

B. Benchmark

The AIM7 benchmark is a well-known benchmark for measuring work throughput. This benchmark provides various

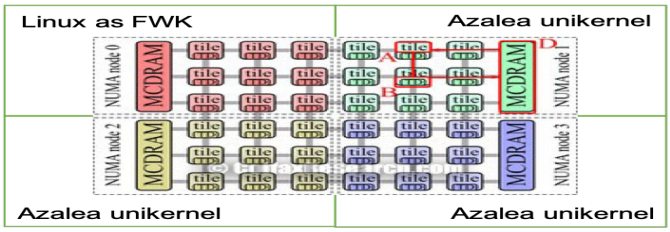

Fig. 3. SNC-4 mode of Xeon Phi and Deployment of Azalea multi-kernel

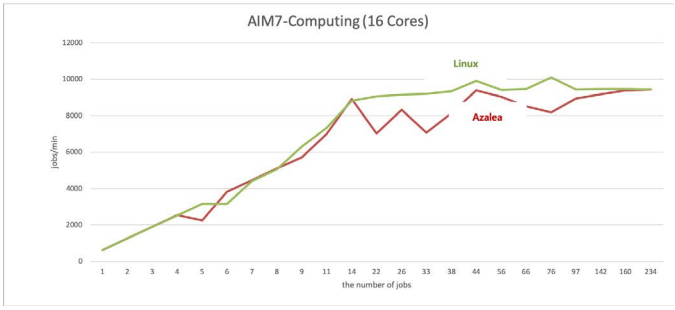

Fig. 4. Performance comparison between Azalea and Linux

workload profile for file services, computation, database, and multi-user to show how well the system performs as needed. In this experiment, we used compute workload to measure the performance of a single unikernel without offloading and used shared workload to measure the scalability of multi-kernel consisting of full weight kernel and unikernel including offloading.

1) compute workload in single Azalea unikernel: Compute workloads consist of tests in which add, div, and multiply operations are performed on data types such as int, short, long, float, and double. The experiment result is shown in Fig. 4. The maximum value of jobs/min on 16 cores indicates that there is little difference between Linux and a Azalea unikernel.

2) multiuser workload in Azalea multi-kernel: To compare Azalea's scalability with Linux, we deployed Azalea multi-kernel as follows. It consists of a full weight kernel to support I / O and three Azalea unikernels to perform AIM7 shared workload. The shared workload consists of 46 tests related to computation, network, file service and etc,. The experiment result is show in Fig. 5. Larger values in this graph mean better scalability. As shown Fig. 5, Linux is more scalable on small cores(<35).But in Manycore (>35), the more increasing the core, the less scalable. In case of Azalea multi-kernel, it can be seen that it has a constant scalability.

V. CONCLUSION

In this paper, we propose Azalea multi-kernel to provide scalability based on a manycore system. The Azalea multi-kernel consists of a full-weight kernel to handle IO and a unikernel to run the application. By separating IO from computation, the performance can be improved by increasing

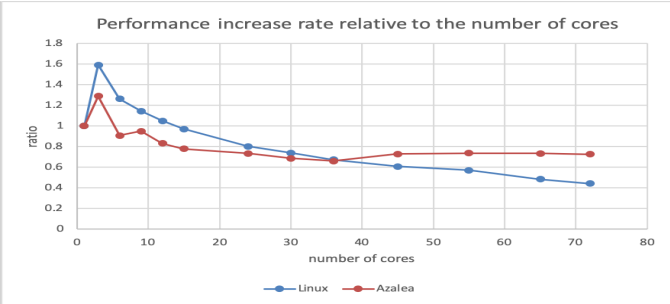

Fig. 5. scalability comparison between Azalea and Linux

the locality of the service. Besides, the performance of the application can be increased by minimizing the noise that may occur in switching between the kernel and user, which is the merit of unikernel. In future work, it will be applied to heterogeneous manycore platforms connected with high-speed interconnects such as Omnipass or InfiniBand.

ACKNOWLEDGMENT

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT)(No.2014-3-00035, Research on High Performance and Scalable Manycore Operating System)

REFERENCES

[1] Hill M D, Marty M. Amdahls law in the multi-core era. IEEE transaction on Computer, 2008.
[2] Baumann, A., Barham, P., Dagand, P.-E., Harris, T., Isaacs, R., Peter, S., Roscoe, T., Schupbach, A., and Singhania A.: The multi-kernel: a new OS architecture for scalable multicore systems. In: 22nd symposium on Operating systems principles, pp. 29–44, Montana (2009).
[3] S.Boyd-Wickizer,H.Chen,R.Chen,Y.Mao,F.Kaashoek,R.Morris, A.Pesterev,L.Stein, M. Wu, Y. D. Y. Zhang, and Z. Zhang. Corey: An operating system for many cores. In Proceedings of the Symposium on Operating Systems Design and Implementation, Dec. 2008.
[4] Andreas Schartl, Design Challenges of Scalable Operating Systems for Many-Core Architecture, 2016
[5] Hillenbrand, M., Bryan, Y.P., Ryu, R.K.D., Bellosa, F.: FusedOS: General-Purpose and Specialized OS Personalities Side by Side. (2013)
[6] Gerofi, B., Ishikawa, Y., Riesen, R., Wisniewski, R. W., Park, Y., Rosenburg, B.: A Multi- Kernel Survey for High-Performance Computing. In: Proceedings of the 6th International Workshop on Runtime and Operating Systems for Supercomputers, pp. 5, Kyoto (2016)
[7] Robert W., Idd I., Parado K., Ravi M., Rolf R. mOS: an architecture for extreme-scale operating systems, In: Proceedings of the 4th International Workshop on Runtime and Operating Systems for Supercomputers (2014).
[8] Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., and Crowcroft, J. Unikernels: Library Operating Systems for the Cloud. In Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems (New York, NY, USA, 2013), ASPLOS 13, ACM, pp. 461472.
[9] Kantee, A. Flexible Operating System Internals The Design and Implementation of the Anykernel and Rump Kernels. PhD thesis, Department of Computer Science and Engineering, Aalto University, Aalto, Finland, 2012.
[10] Bratterud, A., Walla, A., Haugerud, H., Engelstad, and P.E., Begnum, K. IncludeOS: A Resource E cient Unikernel for Cloud Services. In Proceedings of the 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom) (2015).
[11] Lankes, S., Pickartz, S., Breitbart, J.: HermitCore: a unikernel for extreme scale computing. In: Proceedings of the 6th International Workshop on Runtime and Operating Systems for Supercomputers, pp. 4, Kyoto (2016).
[12] Anh, Q., Rukayat, E. David, D. Aravind, P. A Multi-OS Cross-Layer Study of Bloating in User Programs, Kernel and Managed Execution Environments, In: Proceedings of the 2017 Workshop on Forming an Ecosystem Around Software Transformation(2017).
[13] Seung Hyub Jeon, Seung-Jun Cha, Ramneek, Yeon Jeong Jeong, Jin Mee Kim, Sungin Jung. Azalea-Unikernel: Unikernel into Multi-kernel Operating System for Manycore Systems, International Conference on Information and Communication Technology Convergence (ICTC) Transformation(2018).
[14] Embedding with gnu: Newlib, https://sourceware.org/newlib/.
[15] musl-lib, https://www.musl-libc.org.
[16] Pierre Olivier, Daniel Chiba, Stefan Lankes, Changwoo Min, Binoy Ravindran. A binary-compatible unikernel. Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments(2019)
[17] POSIX Threads for embedded systems(PTE),http://pthreads-emb.sourceforge.net.
[18] Intel Xeon Phi Processor 7290F, https://https://ark.intel.com/content/www/us/en/ark/products/95831/intel-xeon-phi-processor-7290f-16gb-1-50-ghz-72-core.html.
[19] Knights Landing (KNL): 2nd Generation Intel Xeon Phi Processor, https://www.alcf.anl.gov/files/HC27.25.710-Knights-Landing-Sodani-Intel.pdf.