Azalea-Unikernel: Unikernel into Multi-kernel Operating System for Manycore Systems

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*—As applications such as a big data processing require more CPUs, manycore systems with a large number of CPUs have been developed to meet this requirement. However, without the in-depth consideration of parallelism, only increasing the number of cores cannot provide performance scalability. Furthermore, the current monolithic operating systems (e.g., Linux) cannot also provide performance scalability in manycore systems due to the cache coherency of shared data problems. As a result, multi-kernel operating systems have emerged as an alternative solution for scalability in manycore systems. We have developed a multi-kernel operating system called Azalea, which consists of a full-weight kernel (FWK) and lightweight kernel (LWK). The FWK handles heavy kernel services (i.e., file system services), and the LWK supports the minimal kernel functions as much as needed of application execution and eliminates the sharing of kernel data. However, there remain kernel noises in LWK such as context switching and page fault handling, even though they are less than Linux. In this paper, we propose Azalea-unikernel, which applies unikernel techniques into the LWK to reduce kernel noises. It eliminates privilege switching and address space switching by integrating user-kernel-address space. In particular case, the azalea-unikernel shows 7.5x better performance than LWK and Linux.**

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

I. Introduction

Applications such as big data processing or in-memory databases require more cores because they demand more and more processing capacity. To meet this requirement, systems consisting of a single node (8 sockets 224 cores), such as Lenovo X3950 or SuperMicro X11 MP, are commercially available. However, there is a limit to increasing the number of cores in a single physical system. Even if the number of cores is increased up to 1000 or more, it is difficult to achieve the expected speedup without extreme parallelism in the operating system(OS) and applications, as explained by Amdahls Law [1].

However, according to extended Amdahls law for manycore systems [2], in the asymmetric environment in which different type of cores coexist, it is possible to guarantee some degree of scalability by performing sequential parts in faster cores and parallel parts in slower cores. Even if these heterogeneous manycore systems are ready, the current monolithic kernel (i.e., Linux) is still an obstacle to performance scalability. For example, Linux has a problem of scalability that does not show comparable performance as the number of cores increase due to cache coherency and data sharing [2] [3] [4]. In order to solve this, various multi-kernel approaches have been proposed.

We developed a multi-kernel operating system called Azalea, which runs on a heterogeneous manycore system for scalability, consisting of multiple nodes, connected using high-speed interconnects [6]. It consists of x86 and knight corners(KNC) and runs multi-kernel that combines lightweight kernel (LWK) in KNC and full weight kernel (FWK) in x86. In addition to lightening the kernel by removing unnecessary elements, the LWK minimizes kernel interference by space-sharing. However, even with this lightweight kernel, application and kernel use different address spaces, resulting in the overhead of replacing address space and performance degradation due to data copy between kernel and user.

In order to overcome this problem, this paper proposes Azalea-unikenel, which applies unikernel techniques to the lightweight kernel in multi-kernel. Unikernel is called a library operating system and can maximize application performance by eliminating mode switching between kernel and user and minimizing data copying by using a single address space [7]. Azalea-unikernel aims to have both advantages - scalability in multi-kernel and performance improvements of application in unikernel, while supporting existing Linux binary compatibility.

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

II. RELATED WORK

A. Multi-Kernel for Manycore

There are various studies on multi-kernel which combines two or more kernels having different roles. FusedOS [8] combines a general-purpose OS (Linux) and specialized OS (CNK). IHK/McKernel [9] uses the proxy model to process LWK requests in the FWK. mOS [10] includes LWK into Linux kernel. Azalea [6] proposes a manycore system and is specialized in I/O.

B. Unikernel

Unikernel can maximize the application performance by minimizing OS noises through a library operating system which only include the necessary service for applications.

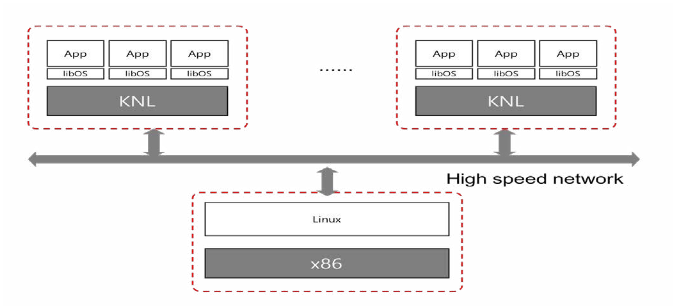

Fig. 1. Manycore System using KNLs and x86

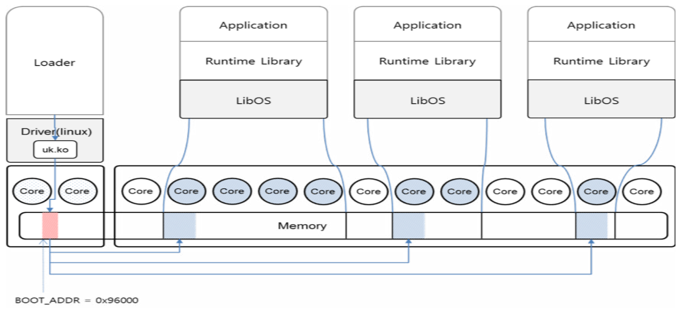

Fig. 2. Azale-unikerenl in a single KNL

There are various unikernels such as MirageOS [7] for various mobile platforms and cloud computing, Rump [11] for enabling POSIX compatible SW, IncludeOS [12] for C++ and Hermitcore [13] for HPC.

III. Azalea-unikernel

Fig.1 shows the heterogeneous manycore hardware platform we are considering. We intend to improve the performance by replacing the Azalea lightweight kernel (Azalea-lwk) with Azalea-unikernel, which has both the merits of lightweight kernel and unikernel. The design goals of Azalea-unikernel are as follows.

- Lighter than lightweight kernel - By allowing applications and the kernel to have a single address space, it is possible to reduce the overhead of changing privileges and associated with TLBs and caches (Fig.3).
- Compatibility - It provides compatibility so that existing legacy applications can be performed. Minimal modification to the original source code or statically built Linux binaries should be allowed to run. To do this, we implement some of the system calls and plan to expand further (Table.I). [14]
- I/O offloading - FWK handles all the I/O so that the application can be executed without interference. I/O performance is improved by direct transfer of I/O data from the device directly to unikernel (Fig.4).

In order to accomplish these goals, the Azalea-unikernel is divided into four major parts based on its functionality (Fig. 2)

A. Sideloader

The sideloader consists of a Linux kernel module and commands for starting and stopping unikernel and reclaiming resources. The sideloader manages the node's resources (CPU, memory) and assigns them to the unikernel so that the application can be executed. In order to provide binary and source compatibility, the sideloader makes it possible to execute not only the binaries linked with the Azalea-unikernel library(libOS) but also the previously statically compiled ELF binaries. Besides, the sideloader handles the system calls requested by the unikernel and transfers the results to the unikernel.

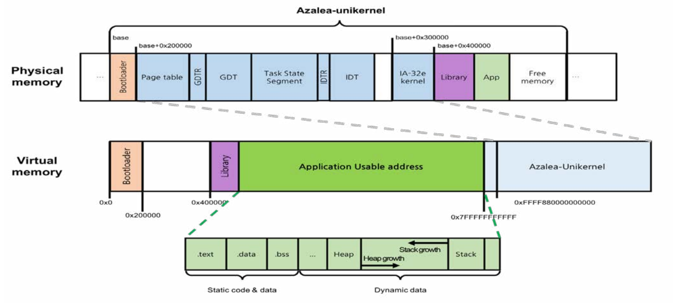

Fig. 3. Physical and Virtual Memory layout for Azalea-unikernel

B. libOS

libOS is a 64-bit simple kernel that provides the OS functions to manage the CPUs and memory allocated from the sideloader. Each core has a queue to manage multi-thread and uses round-robin scheduling. In the case of memory, the kernel and application have a single address space, and Fig 3 shows the physical and virtual memory layout. In order to avoid interrupting the execution of the application, libOS use a per-core data structures and minimize the use of locks.

C. Runtime Library

Azalea-unikernel aims to support glibc, currently supports simple newlib [15], and provides a multi-threading environment that supports some of the pthread api by modifying embedded pthread [16].

D. System Call Handling

Azalea-unikernel supports system calls such as the following list for compatibility with POSIX. System calls are classified into system calls supported internally, and system calls processed through offloading to Linux. Internal system calls correspond to system resources related to threads and virtual memory managed by libOS. Offloading system calls are mainly related to I/O.

To improve the performance of I/O, offloading system call path and data transfer path are separated to minimize data copying. If application calls read (), as shown in Fig 4, libOS forwards this system call to the sideloader and the sideloader does not read the data in the buffer and passes it to libOS and

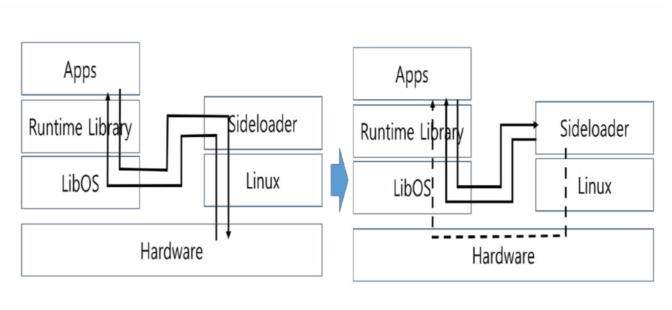

Fig. 4. System call and IO offloading

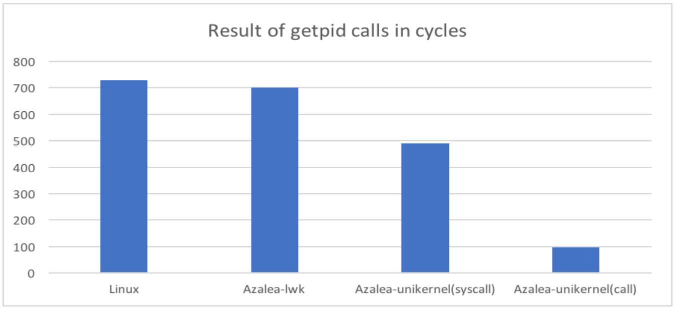

Fig. 5. Results showing comparison of getpid calls in cycles.

TABLE I
LIST OF SUPPORTED SYSTEM CALLS

Category	System Calls
Internal	brk,clone,exit,getpid,clone,mmap,munmap, getprio,setprio,msleep,get_ticks,yield, kill, signal,sem_init,sem_destory,sem_wait, sem_post,sem_timewait,sem_cancelablewait
Offloading	open,close,lseek,lstat,mkdir,read,write,unlink

application directly from hardware. In a single node, shared memory is constructed between sideloader and unikernel. In a manycore connected with multiple nodes using high speed interconnect, we will use RDMA to reduce overhead and latency.

IV. Experimental Results

In this section, we show that Azalea-unikernel is better than Azalea-lwk and Linux through a simple experiment.

A. Testbed

We are aiming for multiple knight landings (KNL) and multiple x86s as the heterogeneous manycore platform for Azalea-unikernel. However, we first use a single KNL as a manycore platform. We use a single Linux as FWK and three unikernels can be run in sub-NUMA cluster modes (SNC-4) of KNL [17], Later, as shown in Fig 1, the manycore system where multiple nodes are connected with the high-speed network will be used as the platform.

B. Results

In order to measure the performance of Azalea-unikernel, the following simple experiment was performed. We consider the average time of 10,000 consecutive calls of getpid, which is the most straightforward system call. Performance is measured in Linux, Azalea-lwk [6], and Azalea-unikernel. In the case of Azalea-unikernel, two different cases are added depending on compatibility level. One is to run a binary, statically built in Linux (SYSCALL), and the other is to execute a binary linked with the Azalea-unikernel library(CALL).

The experiment results are very intuitive, as shown in Fig.5. On Linux and Azalea-lwk, syscall and sysret are called once per each getpid call. On the other hand, for Azalea-unikernel(syscall), only syscall is called. Finally, Azalea-unikernel(call) is treated as call/ret instead of syscall/sysret without mode switches. In this case, Azalea-unikernel shows 1.5x-7.5x better performance than Linux and Azalea-lwk.

V. Conclusion

In this paper, we propose the method of adapting unikernel to the multi-kernel based manycore operating systems. Unikernel holds a single application, and its kernel and application use the same address space. It is possible to reduce the costs of context switching between applications and flushing TLBs when switching between kernel and application. Experimental results show that Azalea-unikernel is lighter than Azalea-lwk and Linux.

In the future work, we will illustrate the performance of Azalea-unikernel through various benchmarks and applications. Moreover, Azalea-unikernel will not only support additional system calls and runtime libraries but also extend to the multi-node environment.

Acknowledgment

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

References

[1] Amdahl. G.M. Validity of the single-processor approach to achieving large scale computing capabilities. In AFIPSConferenceProceedings, vol. 30 (Atlantic City, N.J.. Apr. 18-20). AFIPS Press, Reston. Va., 1967. pp. 483-485.
[2] Hill M D, Marty M. Amdahls law in the multi-core era. IEEE transaction on Computer, 2008.
[3] Baumann, A., Barham, P., Dagand, P.-E., Harris, T., Isaacs, R., Peter, S., Roscoe, T., Schupbach, A., and Singhania A.: The multikernel: a new OS architecture for scalable multicore systems. In: 22nd symposium on Operating systems principles, pp. 29–44, Montana (2009).
[4] 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 Pro- ceedings of the Symposium on Operating Systems Design and Implementation, Dec. 2008.
[5] Andreas Schartl, Design Challenges of Scalable Operating Systems for Many-Core Architecture, 2016

[6] Seung-Jun Cha, Yeonjeong Jeong, Jinmee Kim, Seunghyub Jeon and Sungin Jung : Multi-kernel based Scalable Operating System for Many-core Systems. Advanced Science and Technology Letters Vol.148 (FGIT 2017), pp.28-34

[7] 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.

[8] Hillenbrand, M., Bryan, Y.P., Ryu, R.K.D., Bellosa, F.: FusedOS: General-Purpose and Specialized OS Personalities Side by Side. (2013)

[9] 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)

[10] 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).

[11] 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.

[12] 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).

[13] 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).

[14] 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).

[15] Embedding with gnu: Newlib, https://sourceware.org/newlib/.

[16] POSIX Threads for embedded systems(PTE), http://pthreads-emb.sourceforge.net.

[17] Knights Landing (KNL): 2nd Generation Intel Xeon Phi Processor, https://www.alcf.anl.gov/files/HC27.25.710-Knights-Landing-Sodani-Intel.pdf.