Evaluation of Manycore Operating Systems

Biruk Y. Nidaw
SW Content Research
ETRI
Deajeon Korea
biruky@etri.re.kr

Ziho Shin
SW Content Research
ETRI
Deajeon Korea
zshin@etri.re.kr

Jung Sung In
SW Content Research
ETRI
Deajeon Korea
sijung@etri.re.kr

Kim Young Woo
SW Content Research
ETRI
Deajeon Korea
bartmann@etri.re.kr

Department of Software and Engineering, UST, 217 Gajeong-ro Yuseong-gu, Daejeon, 305350,KOREA
SW Content Research Laboratory ETRI, 218 Gajeong-ro, Yuseong-gu, Daejeon, 34129,KOREA

Abstract—**These days the number of processors in computer systems is increasing in a very fast speed, and different manufacturers are producing computers with manycore processors. In turn, these manycore computer systems require a new operating system that can solve the problems of commodity operating systems. Commodity operating systems originally made for single, dual or multi-cores systems; they will have lower performance and scalability problem for higher number of cores that are hundreds or thousands in number. In this regard, scholars of different research institutions are trying to develop their own solutions for existing problems among commodity operating systems and the recently emerging the manycore processor.**

This paper intends to see what those manycore operating systems are. The main objective of this paper is to compare and contrast some of manycore operating systems, analyze their experimental results, and give some summarizing concepts.

Manycore operating systems are developing from either the scratch or the amendment on the existing commodity operating systems. Operating systems projects that develop a manycore operating system from the scratch, need lots of effort, and generally new approaches are applied. They use completely new technical approaches for development of OSes for increased number of cores. Operating systems like FOS and BarrelFish are some examples of operating systems that developed from the scratch. On the other hand, some operating system projects focus on commodity OSes problems for increased number of cores. These projects amend commodity operating systems and/or used them collectively. Therefore, their main task is preparing systems that play an important role for this amendment. As compared with the above operating system development approach, these approaches are simple and more manycore OSes used it. Operating systems like FusedOS, Cerberus and others are such a type of operating systems.

In this respect, this paper will see the reason behind the selection of either of the two approaches. Also will provide highlights about the basic structural differences among the existing manycore operating systems, in order to catch the philosophy behind them.

Index Terms—**Manycore, operating systems, FOS, Corey, FusedOS, Cerberus, BarrelFish, PopcornLinux,**

I. INTRODUCTION

A. Basics of Operating Systems

Now we are on the era of manycore, where processing performance raising by increasing the number of cores rather than individual clock rates. While there is a great deal of work on programming abstractions for pronouncing parallelism, there is very little work on the operating systems and services that will be needed to support them.

Manycore is result of technology evolution that existed to have solutions for the gap between memories and computation speed; the number of transistors per core increased in order to have better performance, yet the power required for the transistors is not as such small and the instruction level paralleling is not flexible solution in performance upgrading.

Nowadays number of cores per processor increased that did fill those gaps, yet will increase the complexity and scalability problems that let the software will fill these gaps. Let us see:

Power: this portion is the point that shows end of endless scalability. The main reason was power consumption and power dissipation, which their existence resulting from pushing up the frequency of a single core made this path simply not justifiable neither of economic and technological viewpoint. There are numerous factors essential to the CPU power consumption:

$$P_{cpu} = P_{dyn} + P_{sc} + P_{leak} \qquad (1)[2]$$

The dynamic power consumption comes from the activity of logic gates inside a CPU.

$$Pdyn = CV^2Af \qquad (2)[1]$$

Which is equivalent

$$P_{dyn} \cong C.\, V^2.f \qquad (3)$$

Where C = capacitance, V = voltage, A = Activity Factor and f = the switching frequency

Both dynamic and short-circuit power consumption are dependent on the clock frequency, while the leakage current is dependent on the CPU supply voltage.

Communication: In manycore processors, the inter-memory and memory-processor communication will increase exponentially with the increase in the number of cores and respective caches, which needs a software solution in order to have a better performance and minimized latency.

Memory: Program execution, which often limited by the memory bottleneck mainly because of this fact that a significant portion of applications always resides in main memory until it executed by the processor. Multi-core processors can exacerbate problems unless care taken to conserve memory bandwidth and avoid memory contention.

Scheduling: in software design aspect Chip Multi Processers have more challenges than Symmetric multiprocessing which includes, program or thread scheduling and better load distribution on the available cores

In solving these and other manycore processor bottlenecks and further problems, many researchers working late, releasing lots of software solutions, and developing manycore operating systems; the main purpose of this paper is to see the comparison among some of the existing manycore operating systems.

In this paper, some of the well-known manycore operating systems had revealed and presented for further comparison among them in the next chapter. The main impact of this paper is that it enables to provide brief concept on some of the manycore operating systems and give general knowledge about them. From the comparison result, to give a generalized and an immediate information about operating systems in broad and manycore ones at specific.

This research paper organized as follows, in the first part of the paper covers basics and the reason why existence of manycore is indispensable, and then after the research contribution listed. On the second part of this paper broadly known manycore operating system structure and features shown, some additional details for each of those well-known manycore operating systems included. Finally, in this paper comparative analysis and conclusion, which addressed.

II. THE MANY CORE OPERATING SYSTEMS

The Manycore Operating Systems Structures & Features

In frequent manner, noted that manycore revolution is further ahead of the software, which makes the fundamental "technology gaps". It is essential to create bridged before applications can fully take advantage of the massive parallelism and high-performance features of multi-core architectures. In this regard, several manycore operating systems are developing and analyzing for better performance with different features and structures.

TABLE-1 MANYCORE OPERATING SYSTEMS

OS	Developer	Institution
PopcornLinux	Antonio Barbalace et al.	At ECE, Virginia Tech
FOS	David Wentzlaff et al.	CSAI Lab. MIT
FusedOS	Yoonho Park et al	IBM
Cerberus	Xiang S. Haibo et al.	PP Institute, Fudan Univ.
Cory	Silas Boyd-Wickizer et al.	MIT, MS Re. Asia, (Fudan , Xi'an Jiaotong) Univ.
BarrelFish	Adrian Schüpbach et al.	CS Dept. ETH Zürich

A. The Structure perspective

Manycore operating systems either developing from the scratch or making some amendments on the existing commodity operating system in order to overcome the scalability issue.

1) PopcornLinux

PopcornLinux is a Linux based multi- kernel operating system, which targets a platform on which Intel Xeon processors & Intel Xeon-Phi board connected through PCIe. These two different processor cores share access to a common memory. This OS bridges the programmability gap in heterogeneous ISA platforms and provides distributed shared memory transparently to the application. In a multicore box among multi-core processors, one selected as Bootstrap processor that boots up the primary kernel first and secondary kernels and the remaining application processors next.

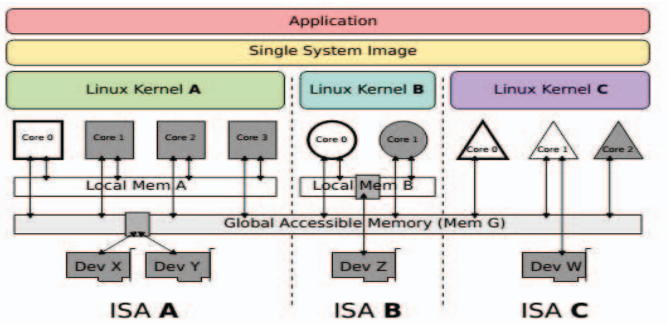

Fig. 1: Heterogeneous-ISA generic hardware model, and Popcorn operating system software layout. [3]

Popcorn can run among different kernels that works on different ISAs. Appears as a single OS and enables the kernels to communicate each other and create global OS through message passing. Its main objective is to hide hardware diversity from application forming an illusion single execution environment.

"Popcorn initializes its namespace that provides a single system image and updates them in every time a new kernel joins. It adds the CPU namespace to Linux that permits an application, to run on a kernel, list the entire kernel and finally migrate to any of them." [3]

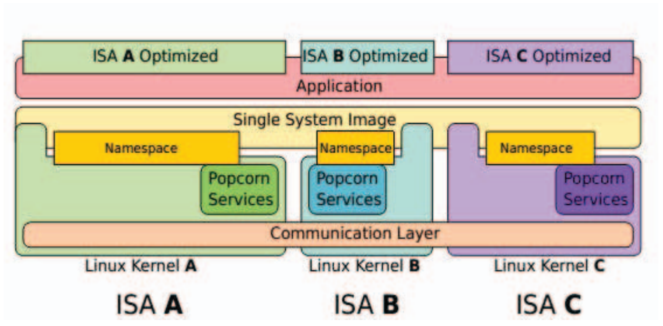

Fig. 2. PopcornLinux kernel- and user-level software layout. [3]

2) Factored Operating System

FOS is a manycore operating system that proposes a solution for hardware locks and that separates the execution of application and operating systems. FOS takes scalability as the first order design constraint, and raises computing problems in regard with an increase the number of cores.

In factored OS design principles, utilization of space multiplexing and factoring the OS for particular service is applied. Its structure has thin micro-kernel, operating system layers (servers), and applications.

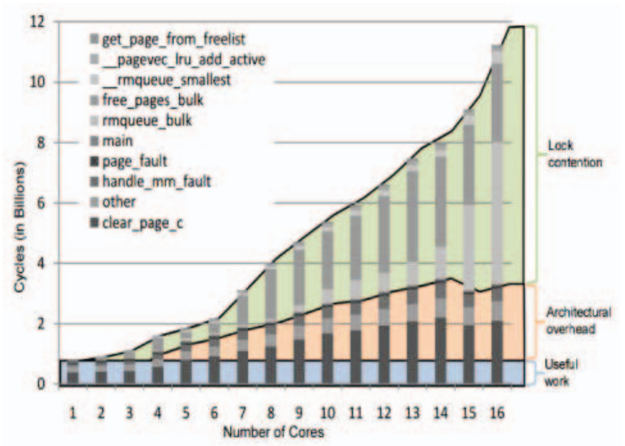

Fig.3. Lock contention of commodity operating system for higher number of cores. [4]

"Application and system execute on separate cores on top of micro-kernel and fleets of function specific servers used to compose operating system layer. FOS microkernel gives reliable messaging layer for microkernel clients." [4]

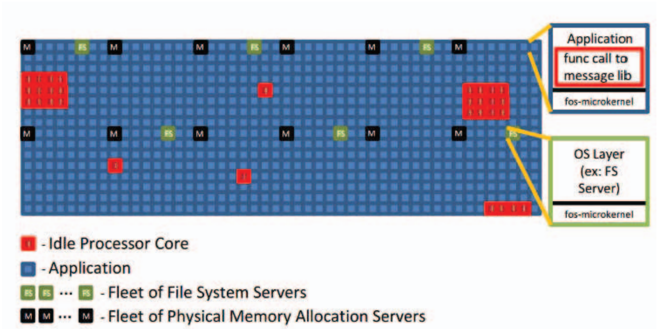

Fig. 4. Operating system and application clients executing on the FOS-microkernel.[4]

Mainly FOS is different from traditional operating systems in lock reduction due to message passing, variation on application server and operating system server, and using message passing.

3) *FusedOS*

FusedOS is an operating system that has both performance of lightweight kernel and functionality of full weight kernel. It makes some modification on Linux as a full weight kernel and implement a user-level lightweight kernel caked compute library (CL)."FusedOS noted the single thread and core power optimization, and designed to optimization of a coherent shared memory between and across STOC (single thread optimized core) and PEC (Power efficient core)."[6] FusedOS achieves LWK for HPC by portioning both core and memory removes OS jitter. In application, perspective FusedOS and Linux will not make conflict with each other. It implemented on Blue Gene/Q.

The three components of FusedOS are PEC monitor, Linux Hooks and PEC FS module and CL. Each component has its own task and interacts in thread and process execution

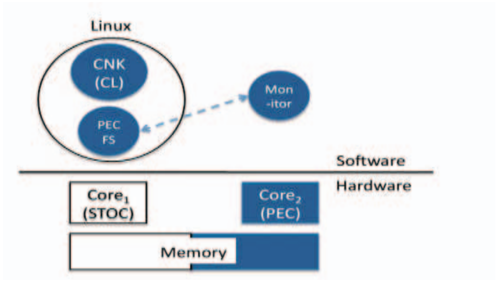

Fig. 5. Components of FusedOS and their structure. [6]

4) *Cerberus*

"Cerberus uses clustering approach, which applies multiple commodity operating systems on top of VMM to serve an application while providing the familiar POSIX programming interface to the shared-memory applications."[7] Its main objective is to make a connection between two manycore OS design approaches, which are designing the operating system from the scratch like FOS, and refining commodity operating systems.

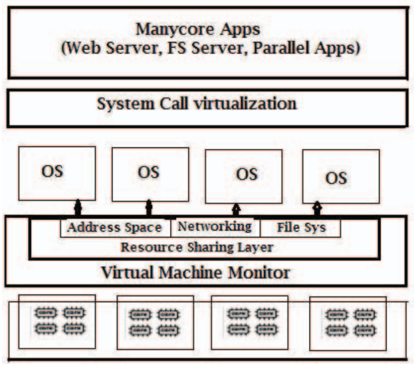

Fig. 6. Architecture overview of Cerberus (Clustering) [7]

Cerberus works on reusing of commodity OSes and includes replication and message passing which are taken as wisdom in new OS design. It also integrates system call virtualization layer that permits an execution of processes or threads in a multiple operating systems. This provides an illusion of running only one operating system for the user.

Cerberus uses SuperProcess notion, which denotes a bunch of processes/threads that executes in multiple operating systems called SubProcess. "It also uses a resource sharing layer in both the VMM and operating systems to support an efficient sharing of resource like file system, networking and address space. For serialization of access to resources among OS instances, Cerberus use message passing and lock free mechanism. VMM controls underlying hardware resources of VMs, one application runs on OS with one master demon and multiple slave demons in various Subprocesses. " [7] The master demon is responsible for loading application initials and creation of demons for slave; finally, it decides which OS to be selected to perform the process/thread creation request in balancing the load.

5) Corey

Corey is inspired by scalability problem of some OS for greater number of core which presents a new technique that overcome memory access bottlenecks by regulating sharing of kernel resources. These goals achieved via permitting concurrent access to common data structure via fine-grain locking and wait-free primitives. It has an approach selective kernel-application communication for shared data assuming that other instances remain private, but it is an incomplete prototype and it cannot be compared with full manycore operating system, rather can be considered as making a case controlling of sharing data between kernel and application.

Corey structured as "exokernel and presented three abstractions called Address Ranges, Kernel cores and Shares." [8] The first abstraction the *address ranges* permit applications to control private or shared address space parts per core. Working on private regions involves no contention, while explicit indication of shared regions allows sharing of hardware page tables and consequent minimization of soft page faults. It also permits applications selectively shares parts of address spaces, instead of forced to make an all-or-nothing decision. The second abstraction *Kernel cores* let applications to dedicate cores, run specific kernel functions, and avoiding contention over the data used by those functions. It manages hardware devices and executes system calls sent from other cores. Kernel core communicates with application cores through shared memory IPC, generally kernel core abstraction enables applications can make decision. Finally, *Shares* are lookup tables for kernel objects that permit applications regulate object identifiers visible to other cores. It lets applications to actively create lookup tables and determine how these tables are shared. Application cores start with one share (its root share), which is private for the core, it is lock free. Note that the three abstractions are implementable without inter-core sharing by default, but consent sharing among cores as directed by applications.

In regard with its evaluation, Corey runs on AMD Opteron and Intel Xeon processors machines. It tested for several benchmarks. The test which, done for each of its abstractions and demonstrated on a MapReduce application which is a combination of Map and Reduce processes.

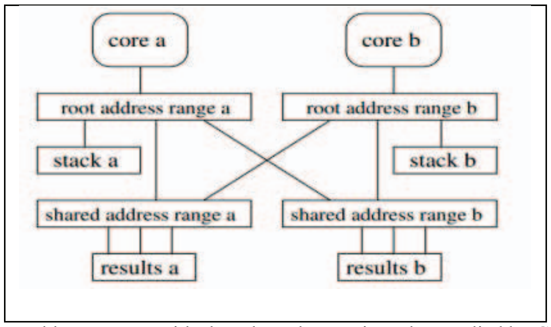

Fig.7. Two address spaces with shared result mappings that applied by Corey [8]

In address ranges regard multicore applications investigated with private and shared memory aspects. To summarize, a Corey application can use address ranges to get good performance for both shared and private memory.

6) BarrelFish

BarrelFish is an OS, which is motivated by both increment in cores and in heterogeneity of systems. The heterogeneity in the context of computing referred to variation in ISA between main processor the rest that have another.

The main objectives are to solve the scalability problems, manage and exploit heterogeneous hardware, and reduce complexity. It categorizes heterogeneity into three,

- Non-uniformity that contains the trend towards non-uniform memory access architecture for scalable multiprocessing,
- Core diversity that denotes heterogeneity of cores in a single system
- System Diversity, which refers existing resources on completely separate systems, are increasingly diverse.

In BarrelFish, sharing substituted with explicit message passing in addition application of sharing and locking between cores for cache level 2 and cache level 3. Note that message passing the better communication system for future hardware system as compared with the sharing.

Application of URPC, which include a fast cross-address space communication protocol using shared memory with lightweight threads used in BarrelFish. This allows the kernel bypassed during cross-address space communication.

For experimental evaluation, BarrelFish achieves higher throughput, less cache misses, and lower interconnect utilization. As it developed from scratch, not tested for scalability. Test suits evaluation done on it and Linux, at this "unfinished BarrelFish OS stage, it achieves almost same result as to that of optimized Linux."[10]

III. Comparative Analysis

Depending the resources of the institutions, those broadly known manycore operating systems performed on different devices and experimental test have done using various benchmarking test suites. In this part of the paper, we put result analysis based on the given manycore operating system experimental results. This enables us to know more about the operating system in terms of the test suite used, the type of device tested on, and other operating systems compared with it; also the test result in comparison with the nominated OS will be given.

PopcornLinux used two E52695 Intel Xeon 12 cores that has two way hyper threaded at 2.4 GHz per socket at dual-socket configuration, 64GB of RAM, and PCIe connected Intel Xeon Phi 57 cores four way hyper threaded at 1.1GHz, 6GB of RAM. Experiment data collected from 8 Xeon cores and 228 Xeon Phi cores, limitation of Xeon is due to NPB (Nas Parallel Benchmarks) that do not show gain in performance for more than 8 Xeon. Note that the fundamental difference between the Xeon and Xeon Phi is, Xeon Phi can run highly parallel code more efficiently than the Xeon processor. On the other hand, Xeon processor run faster on serial and I/O assured workloads. From NPB "Integer Sort (IS), Conjugate Gradient (CG), Embarrassingly parallel (EP) and Scalar penta-diagonal solver (SE) benchmarks" [9] are used for Comparison between PopcornLinux and Native Linux made

To show popcorn works without the need to rewrite an application single system benchmark, the POSIX application

Parallel BZIP version is used. Parallel BZIP creates one thread per processor core available on the platform and two I/O threads. In Popcorn, the number of available cores is returned as the sum of the total cores available on Xeon and the total cores available on Xeon Phi. Therefore, "pbzip2 places its threads on each available core regardless of the CPU architecture. Xeon and Xeon Phi processors have variation in the clock frequency;" [3] due to this popcorn, thread migration does not occur for small number of Xeon Phi.

For the SP benchmark, the partitioner decides always to migrate for 57, 114, and 228 threads, providing improvements over native execution. With 57 threads, Popcorn is up to 53% and 46% faster than native execution on the Xeon Phi and Xeon. For EP benchmarks have the same trend as the SP benchmarks, Popcorn is up to 52% and 53% faster than native execution on the Xeon Phi respectively.

Uniquely to CG, the partitioner migrates for higher number of threads (114 or 228). "In this case, Popcorn is up to 27% and 43% better than native execution on the Xeon."[3] Note here that as Xeon Phi can run highly parallel code more efficiently, native execution on the Xeon Phi has a clear advantage over native execution on the Xeon for only 228 threads.

For IS, which is the fastest benchmark, Popcorn's partitioned IS migrates to Xeon Phi only for 57 and 114 threads. Execution using Popcorn is faster than native execution on the Xeon Phi, also always faster than native execution on the Xeon.

Finally, the native execution time of pbzip2 on the Xeon and the Xeon Phi for different input file sizes Shown on table-3.3. pbzip2 is always faster on the Xeon than the Xeon Phi by a factor of 2, meaning there is no benefit in migrating to the Xeon Phi .

TABLE 2 PBZIP2 EXECUTION TIME ON 8 XEON CORES AND 228 XEON PHI CORES. [3]

File size	128MB	256MB	512MB	1 GB
Xeon 8 cores	4.3 s	8.3s	17.3s	36s
Xeon Phi 228 cores	9s	15s	24s	74s

FOS: from Final Technical Report of FOS for High Assurance and Scalability on Multicores MIT experiments of Factored operating system "FOS is implemented as a paravirtualized OS on Xen to support cloud systems"[5] and compared with native Linux for different benchmarks and it outperforms the native Linux especially for higher number of cores. Fos is implemented several key fleets including the page allocator, name service, network stack, file system, process manager, and cloud manager. FOS supports benchmark workloads like lighttpd, memcached, SQLite, SPLASH, and PARSEC.

The latency of memcached request via FOS and Linux showed that FOS showed good achievement for small workload. The read test and the write test proportionally have same results though the write latency is greater than the read latency.

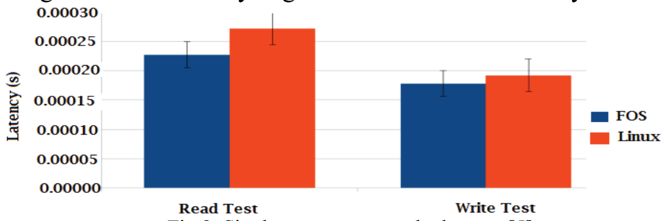

Fig.8. Single stream memcache latency [5]

FOS showed outperform result over Linux on scalability of physical memory allocation service, page allocation, and network stack experimental tests.

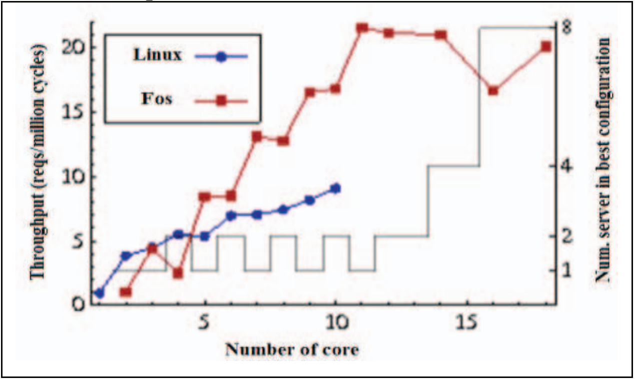

Fig.9. Comparison of FOS and Linux network scaling. [5]

FusedOS evaluation of FusedOS performed on the Blue Gene/Q against operating systems Linux and CNK. "Linux kernel 2.6.32-220.4.2 version from RedHat with some modification by IBM enables to run Blue Gene/Q I/O and compute nodes. Benchmarks used the three Sequoia benchmarks FTQ, STRID3, and LAMMPS"[5] to show that FusedOS provides the same low-noise environment and performance as CNK. Compares ion of FusedOS with Linux, showed that it could potentially provide both LWK performance and FWK features at the same time.

For FTQ benchmark CNK, Linux and FusedOS run tested and large noise spikes showed at the beginning and large amount of frequency as compared with FusedOS and CNK. The output for FusedOS same as CNK with a higher frequency variation which is expected by the developers and assumed due to variation in architectural settings.

For one of the eight STRIDS, STRIDE3, which performed loops of scalar and vector operations, used to test the memory subsystem of a single node of a computational platform for CNK, FusedOS, and Linux. The result is stress on cache, prefetch, and TLB mechanisms; up to STRIDE 455, all showed same performance. Finally, Linux deviated for higher number of STRIDS, but the other two OSes showed same output even for higher number of STRIDS.

For LAMMPS that an HPC benchmark, which mimics "particle like" systems such as molecular dynamics. The comparison among FusedOS, CNK, and Linux, shows FusedOS offers HPC application performance same as to CNK, and it surpasses Linux. FusedOS stimulated, as it does not support MPI. The table-3.1 given below shows Loop time as indicated by LAMMPS, which FusedOS performs 6% lower than CNK and consistently outperforms the Linux.

TABLE 3 LAMMPS PROCESSOR PER THREAD LOOP TIME [6]

Procs x Threads	CNK	FusedOS	Linux
1 x 1	4753.6	4749.7	4877.1
16 x 2	2977.7	3080.1	3258.3
16 x 4	1967.9	2105.3	2453.3
1 x 8	829.0	850.8	890.8
8 x 8	1160.5	1276.8	1446.7
1 x 16	555.7	585.6	599.4
4 x 16	761.7	863.3	972.6
1 x 32	509.5	537.9	564.7
2 x 32	603.2	665.9	744.7
1 x 64	557.1	574.9	669.0

Finally, the system call Latency for null and getpid system calls given as shown on table-4.

TABLE-4 SYSTEM CALL LATENCY FOR NULL AND GETPID FOR FUSEDOS[6]

	Linux	CNK	CL	
			IPI	*waitrsv*
Null	0.0277	0.0275	0.0269	0.0269
Getpid	0.4014	0.2500	8.9082	2.2801

Cerberus: Experimental evaluation performed on "48-core AMD machine with eight 6-core AMD 2.4 GHz, and benchmarks histogram, dbench, Apache web server, and Memcached are used. With OProfile the time distribution of histogram, dbench, Apache and Memcached compared on Cerberus, Xen-Linux and Linux. There are four NIC and each is configured with altered IPs in a subnet."[7]

The performance and scalability of Cerberus compared with native Linux. The performance of Xen-Linux to showed the performance overhead incurred by the virtualization layer, as well as the performance benefit of Cerberus over typical virtualized systems.

Micro-benchmark used to test the time requires to send ping investigate and evaluate performance of Cerberus. The experimental evaluation used 48 core AMD machine for histogram and dbench testes, and 16 core Intel machine for apache webserver and memcached service benchmarks.

For histogram benchmark due to performance overheads Cerberus is significantly worse than Linux for small number of cores, but as the number of cores increased, execution time decreased outperforms native Linux. On 24 core speed of Cerberus is 51% over Xen–Linux. In similar way dbench throughput on Xen-Linux and Linux degraded as the number of core increased from 6 to 12, but Cerberus throughput was worse at the first and better for higher number of cores.

The throughput of Apache server on 16-core Intel machine under the three OSes compered on the per-core throughput of 16 cores, Cerberus showed stable throughput, but Linux at 16-core has 12.1% of single core throughput.

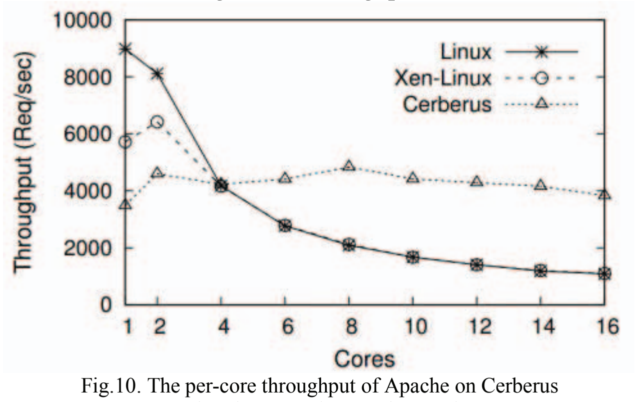

Fig.10. The per-core throughput of Apache on Cerberus compared to those on Linux and Xen-Linux [7]

The Memcached server evaluation on 16-core Intel machine under Linux, en-Linux and Cerberus, have same configuration like Apache web-server.

Corey: All Corey experiments run on "AMD 16-core system with 64 GB of memory, and all Linux experiments use Debian Linux with kernel version 2.6.25 and pin one thread to each core. The kernel is patched with perfctr 2.6.35" [8] to allow application access to hardware event counters. The Streamflow used for Linux MapReduce as provide better performance than other allocators like TCMalloc and gilbc2.7malloc. For network test gigabite switched network used. Corey is expected to have low cost for both cost of private mapping manipulation and the soft page-fault cost that is used in multi-cores. The memclone benchmark is used in resolution of private memory, which assigns all cores in a roundrobin style. Memclone scales well on Corey and on Linux with separate address spaces, but poorly on Linux with a single address space.

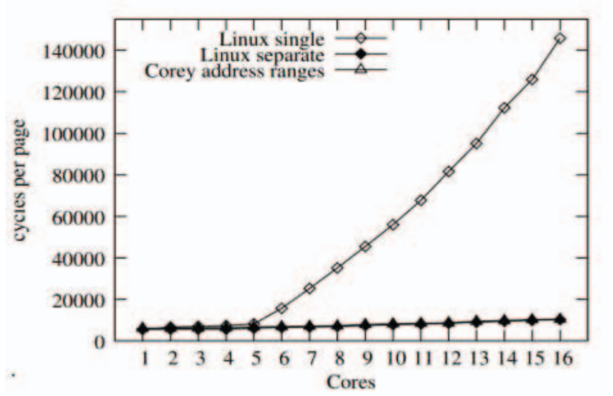

Fig. 11. memclone test result [8]

Mempass is a benchmark that is used to evaluate the costs of memory on multiple cores. It assigns a single 100 MB shared buffer on one of the cores, touches each page of the buffer, and then passes it to another core. The evaluation result showed that Corey and single addressed Linux perform well, while Linux with separate address spaces performs poorly.

The Corey Kernel core abstraction increase the scalability, it is tested simple TCP service benchmark accepting the incoming connections, to each connections writing 128 byte before closing. Two servers with different configurations involved in this test, and one is called "Dedicated" and the second called "Polling". "For Dedicated, each service core uses shared memory IPC to send and receive packet buffers with the kernel core. For Polling, each service core transmits packets and registers receive packet buffers by DMA descriptors (with locking), and is notified of received packets via IPC from the Polling kernel core." [8] The purpose of the comparison is to show the effect on performance of moving all device processing to the Dedicated kernel core.

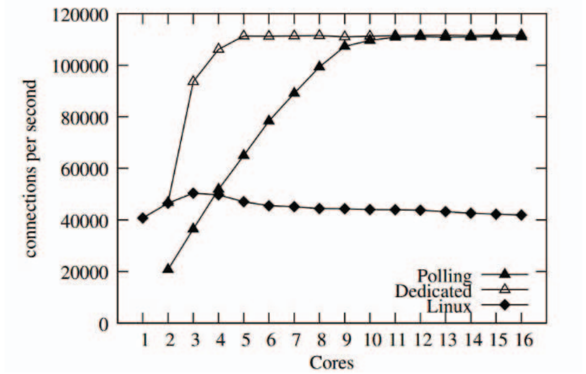

Fig.12. The TCP benchmark [8]

In Corey sharing regard A Corey application can use shares to evade bottlenecks that might be caused by contention on kernel data structures when creating and manipulating kernel objects, that shows Corey's development relative to Linux

BarrelFish experimental evaluation done on "x86-64-based multiprocessors , 2×4-core Intel 2.66GHz Xeon X5355 processors and a single external memory controller. Each processor package contains 2 dies, each with 2 cores and a shared 4MB L2 cache. The 2×2-core AMD system has a Tyan Thunder n6650W board with 2 dual-core 2.8GHz AMD Opteron 2220 processors each with a local memory controller and connected by 2 HyperTransport links. Each core has its own 1MB L2 cache. ," [10]

BarrelFish experiments are evaluated according its objective,

TLB shootdown that is a simplest operation in a multi-processor OS that requires global coordination, In Linux and Window OS IPIs are used and in Barrelfish, messages used for shootdown. The complete message-based unmap operation in Barrelfish quickly outperforms the equivalent IPI-based mechanisms in Linux 2.6.26 and Windows Server 2008 R2 Beta, Enterprise Edition. Look the given on unmap operation.

In addition to this Barrelfish has experimentally evaluated the messaging performance and showed higher throughput, less cache miss rate and lower interconnect utilization as compared with Linux.

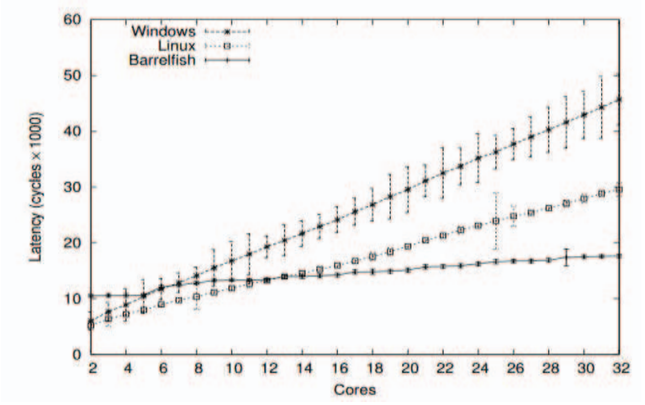

Fig.13. unmap operation comparison [10]

For compute- bound workload the SPLASH-2 benchmark used showed that Barellfish support large shared address space parallel code with little performance penalty. The experiment also show IO workloads that designed to exercise device drivers interrupts and OSes buffering. "The Barrelfish avoid kernel-user crossing by running entirely in user space and communicating over User-level remote procedure call URPC."[10]

In summary of the comparative analysis, we can generalize their experimental analysis with the following table.

TABLE 5 MANYCORE OS EVALUATION SUMMARY

Manycore OS	**Tested on**	**Benchmarks**	**Evaluated with**	**Result**
Popcorn-Linux	Intel Xeon & Xeon-Phi	Nas Parallel Benchmarks	Nat. Linux	Faster than Native Linux
FOS	Xen on cloud sys.	Memcache and Mempass	Nat. Linux	Have Better throughput
FusedOS	Blue - Gene/Q	FTQ, LAMMPS	CNK & Linux	Better HPC performance
Cerberus	48-core AMD	histogram, dbench, Apache web server, and Memcached	Linux Xen-linux	Have Better throughput
Corey	AMD 16-core	Memclone, mempass and TCP service	Si.& sep. Linux	Less contention
Barrel-fish	Intel Xeon X5355	TPC-W and other benchmarks	Linux	Better throughput than both

IV. CONCLUSION

In this research work, we tried to give brief summary of manycore operating systems. In this regard we involved six famous manycore operating systems and non-of them are commercialized; and they are still on their OS advancement.

In their development approach, most of them tried to compare their result in connection with different versions of Linux and showed that commodity operating systems that are made for single, dual or multicore operating systems are not capable to use in manycore system. In conclusion, we did this research as immediate reference for a user in order get what is going in manycore operating system development and re think what should be considered in future manycore OSes development.

V. FUTURE WORK

Electronics and Telecommunication Research institution (ETRI) project named, Research on High Performance and Scalable Manycore Operating System is working on developing manycore operating systems and related aspects in collaboration with other advanced research institutions. Currently this project is developing a new manycore operating system from the scratch and has handled some fundamental steps. In addition to this, it is also developing a Realistic, Cost Effective Testbed for Future Manycore OSes with Xeon Phi, which focus on the experimental evaluation of manycore operating systems by developing conducive testbed on the bases of Xeon phi special behavior. The testbed has features that works on, single-chip manycore system, heterogeneous manycore system and programmable I/O devises. Getting appropriate testbed is one of the difficulty in manycore operating system development.

ACKNOWLEDGMENTS

This research work was fully supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) under contract ETRI MSIP/IITP [B0101-15-0644, Research on High Performance and Scalable Manycore Operating System.

REFERENCE

[1] Binu K. Mathew The Perception Processor http://www.siliconintelligence.com/people/binu/perception/dissertation.html.

[2] CPU power dissipation https://en.wikipedia.org/wiki/CPU_power_dissipation

[3] Antonio Barbalace, Marina Sadini, Saif Ansary, Christopher Jelesnianski, Akshay Ravichandran, Cagil Kendir, Alastair Murray, and Binoy Ravindran Popcorn: Bridging the Programmability Gap in Heterogeneous-ISA Platforms ECE, Virginia Tech

[4] David Wentzlaff and Anant Agarwal Factored Operating Systems (fos): The Case for a Scalable Operating System for Multicores Computer Science and Artificial Intelligence Laboratory Massachusetts Institute of Technology Cambridge, MA 02139

[5] Anant Agarwal, Jason Miller, David Wentzlaff, Harshad Kasture, Nathan Beckmann, Charles Gruenwald III, and Christopher Johnson FOS: A FACTORED OPERATING SYSTEM FOR HIGH ASSURANCE AND SCALABILITY ON MULTICORES Massachusetts Institute of Technology 77 Massachusetts Ave Cambridge, MA 02139-4307

[6] Yoonho Park, Eric Van Hensbergen, Marius Hillenbrand, Todd Inglett, Bryan Rosenburg, Kyung Dong Ryu, Robert W. Wisniewski

FusedOS: Fusing LWK Performance with FWK Functionality in a Heterogeneous Environment 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing

[7] Xiang Song Haibo Chen Rong Chen Yuanxuan Wang Binyu Zang Cerberus A Case for Scaling Applications to Many-core with OS Clustering Parallel Processing Institute, Fudan University

[8] Silas Boyd, Wickizer Haibo, Chen, Rong Chen, Yandong Mao, Frans Kaashoek, Robert Morris, Aleksey Pesterev, Lex Stein, Ming Wu, Yuehua Dai, Yang Zhang, Zheng Zhang Corey: An Operating System for Many Cores *MIT, Fudan University, Microsoft Research Asia, Xi'an Jiaotong University* 8th USENIX Symposium on Operating Systems Design and Implementation

[9] NASA Advanced Supercomputing Division http://www.nas.nasa.gov/publications/npb.html

[10] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter , Timothy Roscoe , Adrian Schüpbach , and Akhilesh SinghaniaThe Multikernel: A new OS architecture for scalable multicore systemsETH Zurich Microsoft Research, Cambridge ENS Cachan Bretagne

Biruk Y. Nidaw received B.S.in Electrical engineering from Adama University in 2010 Adama, Ethiopia. He is assistant lecturer in Adama Science & Technology University, Adama, Oromia Ethiopia. Currently an Integrated Program student in Computer software and engineering Department of University of Science and Technology (UST) in Electronics and Telecommunication Research Institute (ETRI) campus, Deajeon, South Korea.. .His research interest focus on system software, computer architecture and system interconnection design

Ziho Shin received B.S.in Electrical and Electronics Engineering from University of Ulsan in 2015 Ulsan, South Korea. He was worked as research assistant in automatic control lab in Electrical Engineering Department University of Ulsan in 2012 to 2013 Ulsan, South Korea and also he was worked as VRE Operation Intern Goodwill San Francisco|San Mateo|Marine Counties HQ in 2014 San Francisco, California, USA. He is currently enrolled in master course student in Computer software, Department of University of Science and Technology (UST) in Electronics and Telecommunication Research Institute (ETRI) campus, Deajeon, South Korea. His research interest focus on computer architecture, asynchronous circuit design and high-speed system fabric interconnection design.

Sungin Jung received the Ph.D. degree in computer science from ChungNam national university in 2006. He had worked for developing UNIX kernel for SMP, ccNUMA and MPP systems in cooperation with Novell and SCO. In the early 2000s, he was a user and advocate of open source software including Linux in Korea. Currently, he is the director of Basic Research Center for Software, and the project manager of the "Manycore Operating System" national project. His research areas focus on operating system, computer architecture, and cloud computing.

Young Woo Kim received his BS, MS, and PhD in electronics engineering from Korea University, Seoul, Rep. of Korea, in 1994, 1996, and 2001, respectively. In 2001, he joined ETRI, Daejeon, Rep. of Korea. He is an associate professor at the University of Science and Technology (UST), Daejeon, since 2014. His current research interests are high-speed networks and supercomputing system architecture.