Vol II Issue V Nov 2012

Impact Factor : 0.1870

ISSN No :2231-5063

Monthly Multidiciplinary Research Journal

Golden Research Thoughts

IMPACT FACTOR : 0.2105

Welcome to ISRJ

RNI MAHMUL/2011/38595 **ISSN No.2230-7850**

Indian Streams Research Journal is a multidisciplinary research journal, published monthly in English, Hindi & Marathi Language. All research papers submitted to the journal will be double - blind peer reviewed referred by members of the editorial Board readers will include investigator in universities, research institutes government and industry with research interest in the general subjects.

International Advisory Board

Editorial Board

Address:-Ashok Yakkaldevi 258/34, Raviwar Peth, Solapur - 413 005 Maharashtra, India
Cell : 9595 359 435, Ph No: 02172372010 Email: ayisrj@yahoo.in Website: www.isrj.net

Golden Research Thoughts
Volume 2, Issue. 5, Nov. 2012
ISSN:-2231-5063
Available online at www.aygrt.net

ORIGINAL ARTICLE

DESIGN MULTIKERNEL MODEL FOR OPERATING SYSTEM

KAPIL KUMAR

Department of Computer Science,N.M. GOVT. P.G. College, Hansi
Kurukshetra University, Kurukshetra

Abstract:

Previous high-performance computing system have scaled in specific cases but the dynamic nature of modern client and server workload, coupled with the impossibility of statically optimizing an OS for all workloads and hardware variants pose serious challenges for Operating system structures. We investigate a new Operating System structures, the Multikernel that treats the Machine as a network of independents cores, assumes no inter-core sharing at the lowest level, moves traditional OS functionality to distribute system of process that communicate via message-passing. We have implemented a Multikernel os to show that approach is promising and we describe traditional scalability problem for operating system can be effectively recast using message and can exploit insight from distributed system and networking. An evaluation of our prototype on multicore system show that, even on present day macine, the performance of a multikernel is comparable a conventional OS and can scale better to support future hardware. Commodity computer system contain more and more processor cores and exhibit increasingly diverse diverse architectural trade offs, including memory hierarchies, interconnects, instruction set and variants pose serious challenges for operating structures. In our research paper will investigate and propose new OS structure, the multikernel, that treats the lowest level, and moves traditional OS functionality to a distributed system of processes that communicate via message-passing.

KEYWORDS:

Microkernel, multikernel, multi-core processors, inter-process communication.

INTRODUCTION

Computer hardware is challenging and diversifying faster than system software. A diverse mix of cores, caches, interconnects links, input output devices and accelerators, combined with increasing core count, leads to substantial scalability and correctness challenges for OS designers. Such hardware, while in some regards similar to earlier parallel system is new in the general purposes computing system domain. We increasingly find multicore systems in a variety of environment ranging from personal computing platforms to data centers with workloads that are less predicate and often more OS intensive than traditional high-performance computing application. It is no longer acceptable(or useful) to tune a general-purpose OS design for a particular hardware model: the deployed hardware varies wildly and optimizations become obsolete after a few years when new hardware arrives.

Moreover, these optimization involve tradeoffs specific to hardware parameters such as the cache hierarchy, the memory consistency model, and relative costs of local and remote cache access and so are not portable between different hardware types. Often they are not even applicable to future generation of the sane architecture. Typically, because of these difficulties a scalability problem must affect a substantial group of users it will receive developer attention.

We attribute these engineering difficulties to the basic structure of a shared memory kernel with data structure protected by locks, and in this paper we argue for rethinking the structure of the OS as a distributed system of functional units communicating via explicit messages. We identify three design principles:

1.make all inter-core communication explicit
2.make OS structure hardware-neutral
3.view state as replicated instead of shared.

We will develop a multikernel model, which will not only a better match to the underlying hardware(which is networked, heterogeneous and dynamic) but allows us to apply insight from distributed system to the problem of scale, adaptivity, and diversity in operating system for future hardware. Even on present system with efficient cache-coherent shared memory, building an OS using message-based rather than shared data structures, which is limited by the latency of remote data access, the ability to pipeline and batch messages encoding remote operations allows a single core to achieve greater throughput and reduces interconnect utilization. Furthermore the concept naturally accommodates heterogeneous hardware.

DISCUSSION:

Although a new point in the OS design space the multikernel model is related to much previous work on both operating system and distributed system. In 1993 Chaves et al. Examined the tradeoffs between message passing shared data structures for an early multiprocessor, finding a performance tradeoff biased towards message passing for many kernel operations. Machines with heterogeneous cores that communicate using messages have long existed. The Auspex and IBM system/360 hardware consisted of heterogeneous cores with partially shared memory, and unsurprisingly their OSes resembled distributed system in some respect. What is new in the scale pf parallelism and the diversity of different machines on which a general-purpose OS must run. Similarly, explicit communication has been used on large-scale multiprocessor such as the cray T3 or IBM Blue Gene, to enable scalability beyond the limits of cache-coherence. The problem of scheduling computation on multiple cores that have the same ISA but different performance tradeoffs is being addresses by the Cypress project. Also related is the fos system which targets scalability through space-sharing of resoueces. Most work on Os scalability for multiprocessor to date has focused on performance optimizartions that reduce sharing. Tornado and K42 introduced clustered objects, which optimize shared data through the use partitioning and replication. However the base case and the mean by which replicas communicate, remains shared data. Similarly, Corey advocate reduceing sharing witin th Os by allowing applications to specify sharing requirements for Os data, effectively relaxing the consistency of specific objects. As in K42 however the base case for communication is shared memory. We will design and construct the OS as a shared-nothing distributed system which may locally share data(transparently to application) as an optimization. We see a multikernel as distinct from a microkernel, which also uses message-based communication between processes to achieve protection and isolation but remains a shared-memory multithreaded system in the kernel. For instance Barrelfish has some structural similarity to a microkernel in that it consist of a distributed system of communicating user-space processes which provide services to applications. However unlike multiprocessor microkernel, each core in the machine is managed completely independently. The CPU driver and monitor share no data structures with cores expect for message channel. That said some work in scaling microkernels is related. The microkernel comparison is also informative. Disco and Cellular were based on the premises that large multiprocessor can be better programmed as distributed system. Prior work on" distributed operating system" aimed to build a uniform os from a collection of independent computer linked by a network. There are obvious parallels with the microkernel approach, which seeks to build an OS froma collection of cores communicating over links within a machine, but also important differences:

1.A multikernel may exploit reliable in order message delivery to substantially simplify its communication.
2.The latencies of intra-machine links are lower(and less variable)than between machines.

In this paper we sketch out the multikernel architecture for an operating system built from the ground up as a distributed system, targeting modern multicore processors intelligent peripherals, and heterogeneous multiprocessors, and incorporating the idea above. We don't believe that this is the only'or even necessarily the best, structure for such a system. However it is useful for two related reason. Firstly it represents an extreme point in the design space, and hence serves as a useful vehicle to investigate the full consequences of viewing the machine as a networked system. Secondly, it is designed with total disregard for compatibility with either windows and Posix. In practice we can achieve compatibility with sub optimal performance by running a VMM over OS and this gives us the freedom to investigate OS APIs better suited to both modern hardware and the scheduling and I/O requirements of concurrent language runtimes.

CONCLUSSIONS:

We do not advocate blindly importing distributed systems ideas into OS design, and applying such ideas usefully in an OS is rarely straightforward. However, many of challenges lead to their own interesting research direction.Modern computers are inherently distributed systems, and we miss opportunities to tackle to the OS challenges of new hardware if we ignore insights from distributed systems research. We have tried to come out of denial by applying the resulting ideas to a new OS architecture the multikernel.

REFERENCES:

[1] E. Alkassar, N. Schirmer, and A. Starostin. *Formal pervasiveveri_cation of a paging mechanism. In C. R. RamakrishnanJ. Rehof, editors, TACAS, volume 4963 of LNCS, pages109{123. Springer, 2008.*

[2] J. B. Dennis and E. C. Van Horn. *Programming semanticsfor multiprogrammed computations. CACM, 9:143 I 155,1966.*

[3] D. Elkaduwe, G. Klein, and K. Elphinstone. *Veri_edprotection model of the seL4 microkernel. In J. Woodcockand N. Shankar, editors, VSTTE 2008 | Veri_ed Softw.:*Theories, Tools & Experiments, volume 5295 of LNCS,pages 99 I 114, Toronto, Canada, Oct 2008. Springer.

[4] I*SO/IEC. Programming languages | C. Technical Report9899*:TC2, ISO/IEC JTC1/SC22/WG14, May 2005.

[5] X. Leroy. *Formal certi_cation of a compiler back-end, or:Programming a compiler with a proof assistant. In J. G.Morrisett and S. L. P. Jones, editors, 33rd POPL, pages42{54, New York, NY, USA, 2006. ACM.*

[6] J. Liedtke. *Towards real microkernels. CACM, 39(9):70{77,Sep 1996.*

[7] Z. Ni, D. Yu, and Z. Shao. Using XCAP to certify realisticsystem code: Machine context management. In 20th TPHOLs, volume 4732 of LNCS, pages 189{206,, Germany, Sep 2007. Springer.

[8] T. Nipkow, L. Paulson, and M. Wenzel. Isabelle/HOL | A Proof Assistant for Higher-Order Logic, volume 2283 of LNCS. Springer, 2002.

[9] AMD. AMD64 *Architecture Programmer's Manual Volume2: System Programming, Sept. 2007. Publication number 24593.*

[10] C. Amza, A. L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, W. Yu, and W. *Zwaenepoel. TreadMarks:Shared memory computing on networks of workstations.IEEE Computer, 29(2), 1996.*

[11] T. E. Anderson, B. N. Bershad, E. D. Lazoswka, and H. M. Levy. *Scheduler activations: E_ective kernel support for the user-level management of threads. ACM* Transactions on Computer Systems, 10:53–79, 1992.

[12] J. Appavoo, D. Da Silva, O. Krieger, M. Auslander, M. Ostrowski, B. Rosenburg, A. Waterland, R. W. Wisniewski, J. Xenidis, M. Stumm, and L. Soares. *Experience distributing objects in an SMMP OS. ACM Transactions on Computer Systems, 25(3), 2007.*

[13] K. R. Apt and M. G.Wallace. *Constraint Logic Programming using ECLiPSe. Cambridge University Press, 2007.*

[14] H. E. Bal, R. Bhoedjang, R. Hofman, C. Jacobs, K. Langendoen, T. Rühl, and M. F. Kaashoek. *Performance evaluation of the Orca shared-object system. ACM Transactions on Computer Systems, 16(1):1–40, 1998.*

[15] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A.*Warfield. Xen and the art of virtualization. In Proceedings of the 19th ACM Symposium on Operating Systems Principles, pages 164– 177, Oct. 2003*

[16] H. M. Mathis, A. E. Mericas, J. D. McCalpin, R. J. Eickemeyer, and S. R. Kunkel*, "Characterization of simultaneous multithreading (SMT) efficiency in POWER5,"IBM J. Res. Dev., vol. 49, no. 4/5, pp. 555–564, 2005.*

[17] J. M. Borkenhagen, R. J. Eickemeyer, R. N. Kalla, and S. R. Kunkel, *"A multithreaded PowerPC processor for commercial servers.," IBM Journal of Research and Development, vol. 44, no. 6, pp. 885–898, 2000.*

[18] S. Storino, R. Eickemeyer, R. Kalla, and S. Kunkel, *"A commercial multithreaded RISC processor," Digest of Papers, International Solid-state Circuits Conference,pp. 236–237, 1998.*

[19] J. M. Tendler, J. S. Dodson, J. S. F. Jr., H. Le, and B. Sinharoy, *"POWER4 system microarchitecture,"*

IBM Journal of Research and Development, vol. 46, no. 1, pp. 5– 26, 2002.
[20] Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands,P., Keutzer, K., Patterson, D. A., Plishker,W. L., Shalf, J., Williams, S. W., and Yelick, K. A. *The landscape of parallel computing research: A view from Berkeley. Tech. Rep.UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.*
[21] Baumann, A., Barham, P., Dagand, P., and T. *The Multikernel: A new OS architecture for scalable multicore systems. In SOSP'09.*
[22] Blackburn, S. M., Garner, R., Hoffman, C., Khan, A. M., McKinley, K. S., Bentzur, R., Diwan, A., Feinberg, D., et al. *The DaCapo benchmarks: Java benchmarking development and analysis. In OOPSLA '06.*
[23] Boyd-Wickizer, S., Chen, H., Chen, R., Mao, Y., Kaashoek,M. F., Morris, R., Pesterev, A., Stein, L., Wu, M., hua Dai, Y., Zhang, Y., and Zhang, Z. Corey: *An operating system for many cores. In OSDI'08.*
[24] Boyd-wickizer, S., Clements, A. T., Mao, Y., Pesterev, *A., Kaashoek, M. F., Morris, R., and Zeldovich, N. An analysisof Linux scalability to many cores. In OSDI'10.*