Paper Review Form (1000 words maximum)
cs940: Xen and the Art of Virtualization [1]

January 28, 2018

Paper Summary

3–5 sentences. Briefly summarise the **contributions** *of the paper, i.e., what it adds over the state of the art. Paraphrase and extract the essentials rather than simply copying chunks of text. Be objective; later sections allow for your own opinion.*

Built on the state-of-art of virtualisation systems at the time, Xen is a virtual machine monitor supporting a near-universal range of modified guest operating systems in full isolation, through the use of paravirtualization (PV). Designed for x86 but with extensibility for porting to other architectures, Xen operates as an efficient intermediary between hardware and fully-featured guest systems. Xen allows all guest operating system features to operate in a similar way as if on physical hardware, but under security mechanisms enabling guest isolation and misuse protection. Xen's performance and scalability were evaluated to be superior to all alternative solutions of the time.

Pros and Cons

6 bullets. Succinctly state three positives and three negatives of the paper.

I believe that the paper has the following positive features:

- Without processor virtualization support, the paper presented a solid case for implementing paravirtualization at the time, with a comprehensive set of design principles later proved to be highly influential.
- As a virtualisation solution that requires guest system modification, the authors covered their porting effort of a wide range of popular operating systems in detail, closely connected to Xen's features.
- In addition to evaluation through conventional benchmarks, the authors demonstrated the performance of Xen under extreme scenarios, such as under very high level of scaling and malicious guest systems.

I believe that the paper has the following negative features:

- The solution was tailored towards minimising overhead on x86 systems, with some assumptions made on optimisations in modifications made on guest kernels, which may not be possible on all guest kernels or for all architectures.
- It was evident that some systems were more complex to port to Xen than the authors expected, especially proprietary systems such as Windows, raising the possibility of complexity bias towards the open-source Linux.
- It was demonstrated that spreading heavy multiprocessing over Xen guests results in a better performance than multiprocessing on a single Linux system. However, the same effect on Xen's alternatives at the time were not evaluated, making it uncertain whether alternative solutions could perform better than Xen in this aspect.

The Problem/Motivation

1–2 sentences per question. What is the motivation for the work, or the problem being solved? Why is it important? If there is prior art, how was it insufficient? If the problem had not previously been solved, why not?

Virtualization of system resources had long been an approach to achieve independent executions of applications for various security and performance reasons [2]. A majority of operational solutions at the time involved full virtualization of the underlying hardware [3] [4], which had severe penalties due to lack of hardware virtualization support at the time, as demonstrated later by the authors' relative evaluations.

The alternative approach of paravirtualization – modifying guest system kernels to support low-overhead full functionality virtualization – was in its early state of development, with existing solutions lacking in multi-application guest system support [5] or safety in executing unverified guest applications [6]. Xen seeks to bring security and fairness assurances to resolve these drawbacks in paravirtualization.

The Solution/Approach

5–10 sentences. What have they done? How does it address the issues set out above? How is it unique and/or innovative (if, indeed, it is)? Give details, again using the paper as the source but again, not just copying text. Instead, focus on paraphrasing/synopsising, and extracting the essential details.

Under a set of design principles, Xen seeks to utilise advantages brought by paravirtualization to support a variety of guest operating systems with minimal kernel modification cost, and to execute guests in full isolation with minimal overhead introduced by virtualization. In conjunction, these features overcame the drawbacks faced by paravirtualization systems at the time.

In particular, Xen preserves the flexibility of guest-managed TLBs by reserving a small section in every address space, while verifying all updates to page tables created by guests, with batched updates available to improve performance. On x86, Xen PV utilises the unused privilege ring 1 to execute guest kernels, for the protections of both Xen from guest kernels, and guest kernels from their ring 3 applications. Xen virtualises privileged instructions and exceptions, with fast exception handlers available to improve guest performance. Asynchronous I/O rings were implemented for fast I/O between Xen and guests. Xen also safely exposes hardware details such as hardware page locations and the real time to guests for performance optimisation. Live guest management functionalities are also available in Xen to allow flexible guest deployment.

Evaluation

3–4 sentences. How do they evaluate their work? What questions does their evaluation set out to answer? What does their evaluation say about the strengths and weaknesses of their system? What is your opinion of the strengths and weaknesses of the evaluation itself? Give highlights, not a point by point reproduction of the evaluation section(s). In rare cases, systems papers may not have any evaluation, in which case write 'N/A' below.

In a standard systems research approach, the authors evaluated Xen against its alternative solutions with a range of microbenchmarks on specific functionalities of interest, and general benchmarks under more realistic performance-critical scenarios. Reportable benchmarks painted a comprehensive picture of Xen's performance.

Under the common context of single-processor support at the time, Xen achieved superior relative performance in CPU-bound and I/O-bound benchmarks. In general OS benchmarks, Xen exhibited a small increase in latency and minor network performance overhead in comparison with bare-metal Linux, but significantly better than full virtualization alternatives. While the penalty effect of paravirtualization overhead was evident when executing a single process, isolating application instances in individual guest systems proved to scale better than single-system multiprocessing. Through scheduling fairness, Xen also achieved excellent isolation from disruptive guests and scalability when hosting a large number of guests.

Your Opinion

At least 3 sentences. This is the fun part where you get to judge both the paper and the work it reports! Is the motivation convincing? The problem important? The approach a good or bad idea? Why? Which specific things annoyed you, or you thought were cool, or cool-but-flawed? Justify your opinions! Make an argument which will convince others of your opinion.

Before main CPU manufacturers started to provide hardware virtualization support, Xen paravirtualization was the product of the time that best utilised existing architecture and hardware features to provide low-overhead virtualization. The development of fully guest isolation also led to a major shift in the primary market of virtualization – from efficient application provisioning to the virtual machine hosting market, later coined with the term *cloud*. Xen PV was the de facto standard of virtualization for these services (e.g. Amazon

AWS and Linode) until Intel and AMD provided hardware virtualization support in their respective processors (VT-x [7] and AMD-V).

For Intel platforms, VT-x significantly reduced the need for paravirtualization by allowing the hypervisor to execute in a special root-mode underneath ring 0, on which fully-virtualised guest kernels can directly operate. The performance benefit brought by paravirtualization became outweighed by its inflexible kernel modification requirement. Xen PV has been gradually phased out, in favour of its full virtualization variant HVM, and its alternative Linux KVM [8].

Questions for the Authors

Finally, imagine you're attending a talk about this paper given by one of the authors. Give at least 2 questions that you would like to ask, specific to the paper and the research it reports.

- When executing a large number of processes in individual guests, Xen achieves significantly better performance than single-system multiprocessing due to scheduling fairness and guest isolation. However, guest isolation also removes their ability to communicate efficiently through shared memory. Do you think that under this particular use case, message-passing based solutions with specialised runtime (e.g. Mirage [9]) would eventually replace Xen?
- What foresight in relation to the VM hosting market have you considered while designing Xen? What features of Xen do you think that enabled Xen's rapid popularisation in the hosting market, or that eventually led to Xen being phased out in favour of alternatives like KVM [10]?

References

[1] 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 *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 164–177.

[2] Y. Xing and Y. Zhan, "Virtualization and cloud computing," in *Future Wireless Networks and Information Systems*. Springer, 2012, pp. 305–312.

[3] G. D. Goud, V. J. Zimmer, and M. A. Rothman, "Method and apparatus for providing virtual server blades," Apr. 6 2010, uS Patent 7,694,298.

[4] S. W. Devine, E. Bugnion, and M. Rosenblum, "Virtualization system including a virtual machine monitor for a computer with a segmented architecture," May 2002, uS Patent 6,397,242.

[5] A. Whitaker, M. Shaw, and S. D. Gribble, "Scale and performance in the denali isolation kernel," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 195–209, 2002.

[6] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 59–64, 2003.

[7] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith, "Intel virtualization technology," *Computer*, vol. 38, no. 5, pp. 48–56, 2005.

[8] C. D. Graziano, "A performance analysis of xen and kvm hypervisors for hosting the xen worlds project," 2011.

[9] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, “Unikernels: Library operating systems for the cloud,” in *ACM SIGPLAN Notices*, vol. 48, no. 4. ACM, 2013, pp. 461–472.

[10] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, “kvm: the linux virtual machine monitor,” in *Proceedings of the Linux symposium*, vol. 1, 2007, pp. 225–230.