Paper title: Xen and the Art of Virtualization
Paper authors: P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield
Your name: Parth Kansara (115135130)

What problem does the paper address? How does it relate to and improve upon previous work in its domain?
The paper addresses the various issues related to virtualization - the need for additional hardware, a lack of support for commodity OS, resource isolation or performance guarantees, along with tradeoffs between binary compatibility & performance, security or functionality & speed.

To address the lack of performance isolation in a conventional virtualization system, additional support can be added as in **Linux/RK**, **QLinux** and **SILK.** However, these systems are overridden by the problem of *QoS crosstalk*, which can be resolved by lowering the multiplexing as much as possible. Further, the x86 architecture lacks support for full virtualization. **VMware's ESX Server** performs virtualization by adding traps to the guest OS, along with maintaining shadow page tables. This has high costs in update-intensive operations. The **Denali project** supports thousands of VMs running network services, unlike Xen, which allows 100 VMs running standard apps and services. In terms of design, the Denali project does not target existing ABIs and does not address the problem of application multiplexing or multiple address spaces, within the guest OS. Also, it requires the VMM to perform all the paging due to lack of support for memory management at the virtualization layer and ensures protection by virtualizing the namespaces of all machine resources.

Several systems including **IBM VM/370**, **VMware** & **Connectix** leverage full virtualization, despite its shortcomings. **Disco** uses the VMM approach to run commodity OS on ccNUMA machines. **IBM** supports paravirtualized Linux for their zSeries mainframes. **vMatrix** and **IBM's Managed Hosting service** both use low-level virtualization for distributed systems. **The PlanetLab** provides a system for research of distributed network services using **VServers** and **SILK** to govern the sharing. The **Xen architecture** is inspired from these communities and attempts to provide a generalized solution for virtualization.

What are the key contributions of the paper?
The paper presents a VMM for the x86 architecture, called Xen which hosts commodity OS while sharing the hardware and upholding the performance. Xen allows users to run untouched application binaries in a resource-controlled manner, along with providing a high level of flexibility and performance isolation. To circumvent the shortcomings of full virtualization and offer superior performance along with resource isolation, the paper provides an abstraction of the underlying hardware that is not exactly identical, in an approach called paravirtualization.

In terms of memory management, x86 does not have a software-managed TLB so Xen requires the guest OSes to allocate and manage the hardware page tables. Xen occupies a 64MB section at the top of every address space so as to prevent TLB flushing whenever it switches to the hypervisor. The guest OS registers the allocated page with Xen, every time a new page table is required. Post that, Xen validates every update request sent by the guest OS via hypercalls, which may further be batched to amortize the switching overhead. For virtualizing the CPU on x86, Xen is made to execute in ring 0 while the guest OS runs in ring 1 and guest applications in ring 3. Exceptions are virtualized by registering a table of each type of exception handler with Xen, which can be used by Xen's handler to create a copy of the exception stack on the guest OS and cede control to the appropriate registered handler. System call exceptions are further optimized by permitting each guest OS to provide a *fast* exception handler, which can be directly accessed by the processor. For virtualizing I/O devices, Xen transfers the I/O data via shared-memory, asynchronous buffer-descriptor rings. Further, it also has an event-delivery mechanism to notify the domain asynchronously. Xen utilizes the Borrowed Virtual Time (BVT) algorithm to schedule domains, as it is work-conserving and has a unique mechanism for fast dispatch of a domain, which favors recently woken up domains. The guest OSes are also offered 3 ideas of time - real time, virtual time and wall-clock time.

Briefly describe how the paper's experimental methodology supports the paper's conclusions.
The paper compares the performance of XenoLinux, with native Linux and other VMMs. XenoLinux bears a meager 3% overhead as compared to the native Linux, while other VMMs show a significant slowdown. For the SPEC WEB99 benchmark, XenoLinux performs within 1% of the native Linux's performance, while VMware and UML both support less than one-third of the native Linux's clients. For the process microbenchmarks, Xen performs slower than native Linux due to verification of every update by Xen, but this overhead cost can be amortized by batching. In case of a malicious workload, Xen is able to achieve a high degree of performance isolation, facing a 4% and a 2% slowdown on OSDB-IR and SPEC WEB99 benchmarks. Lastly, Xen is able to host 128 domains with only a 7.5% drop in total throughput compared to Linux. This demonstrates the efficacy of Xen in hosting up to 100 virtual machines without sacrificing performance
Write down one question you may want to ask the authors.
Could Xen benefit from the hardware support introduced by Intel and AMD for virtualization?