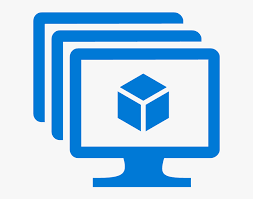
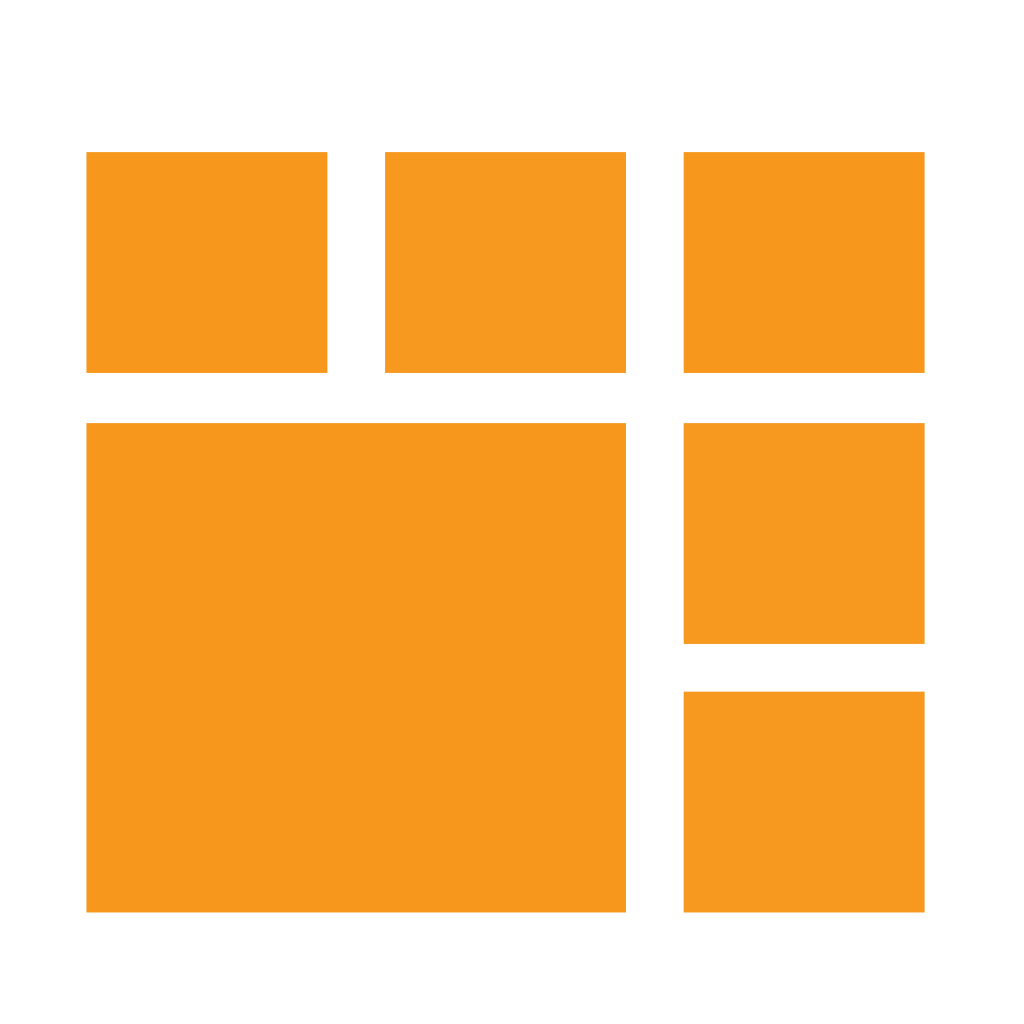
Review on

Virtualization Papers

By Deepak Kasi Nathan



**V**

irtualization of modern commodity OS is playing important role in computing these days but historically this has been very hard thing to do as we had several software and hardware architecture constraints but as time progressed people came with different novel solutions that took us one step ahead . Few part of that big story has been discussed here . Let's dive into few advancements of last two decade that brought us here today in virtual machine space .

# 1 Introduction

Virtualization is an age old topic that’s been lingering around computer engineering field for more than half a century .In early 1970s , Popek and Goldberg suggested three guidelines for necessary for virtualizing such as fidelity , performance and safety . During those times only one style of VMM implementation style, trap and emulate, was the only feasible solution . Since then we have been reliant on that method as we need some kind of VM and also use cases expanded widely to different needs , we had so many techniques to virtualize computing resource as we saw some required specialized hardware, or cannot support commodity operating sys- tems , few targeted 100% binary compatibility at the expense of performance whereas others sacrifice security or functionality for speed. In Early 2000s we had Xen, x86 virtual machine monitor which allows multiple commodity operating systems to share conventional hardware without sacrificing performance or features. Which was achieved by providing an idealized virtual machine abstraction to which operating systems were ported with minimal effort to offer features such as server consolidation, co-located hosting facilities , distributed web services , secure computing platforms and application mobility and also Xen retrofitted support for performance isolation to the operating system .

Support for full virtualization was never part of the x86 architectural design, That’s why VMMs like Xen were so important at that time but as time progressed , demand for hardware assisted vitalization grew popular, as x86 lacked hardware support for virtualization ,while **paravirtualization** has produced promising results, but they are not always practical or feasible . Binary translation was used to fully virtualize x86 at that time ,when x86 hardware manufacturers announced architectural extensions to support virtualization natively in hardware. This support to for existing software based techniques to use hardware natively had opened multiple avenues to optimize performance and get desired results for next decade .

In 2020 , Cloud service providers allow end-users easy access to se- cure, elastic and state-of-the-art resources, with efficient management techniques .Resource *virtualization* is used to maximize the utilization of the hardware that’s serving computing resources . One such component in the cloud stack which can be very crucial is the *Virtual Machine (VM) allocator*, which assigns VM requests to the physical hardware. Bad or inefficient VM placement decisions can result in fragmentation (as even 1 percent of fragmentation can lead upto 100 million dollars in cost), performance reduction and service delays, some cases even rejection of incoming requests and that impacts customer adversely . Hence Azure wanted their VM allocator to robust , extensible , flexible and highly optimized .

To achieve the desired metrics for such large inventories like Azure , Protean’s design has distinct separation between *policy* and system *mechanisms*. Policy is expressed through a flexible rule-based Allocation Agent (AA), which addresses numerous constraints and performance criteria for allocating VMs. On the system side, a multilayer caching keeps track of previous allocation outcomes through efficient update mechanisms,. The memory footprint of the cache is easily manageable (e.g., around 1GB for 10k machines), and scales sub-linearly for additional machines. Protean can easily serve the peak demands and also provide sustained performance as it is showing its capability during the COVID-19.

The paper called “Xen and the Art of Virtualization ” by Paul Barham∗, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer†, Ian Pratt, Andrew Warfield neatly describes how folks at University of Cambridge worked on creating Xen , where as “A Comparison of Software and Hardware Techniques for x86 Virtualization “ per by folks at VMware (Keith Adams and Ole Agesen ) dealt with hardware virtualization and finally “Protean: VM Allocation Service at Scale “ by Microsoft research team explains aspects VM resource allocator .

This review paper begins with Overview (§2) of Xen , software & hardware visualizations and Protean where we discuss how differently they are envisioned , with brief notes on their goals at that point. We then discuss the performance of all systems (§3), Finally, we have a glimpse of future on virtual machines(§6), and conclude.

# 2 APPROACH & OVERVIEW

# 2.1 Xen Overview

As we have said above , support for full virtualization was never part of the x86 design. So few instructions must be handled by the VMM for good support of virtualization, but these instructions fails without raising any bell instead of causing trap , these are solvable problems but with sacrifice on performance. For example VMware’s ESX Server dynamically rewrites hosted machine codes to insert traps wherever VMM intervention might be required based on speculation .

We could have avoided the drawbacks of full virtualization by means of *paravirtualization*. This promised improved performance, although it does require modifications to the guest operating system but no modifications are necessary for guest *applications*. Ultimately the design of Xen was based on few design principles that needs to be addressed . First of it would be to Support full multi-application operating systems, second would be Paravirtualize to support natively uncooperative machine architectures such as x86 and finally to hide effects of resource virtualization from guest OSes . In the following sections we will see how virtual machine abstraction exported by Xen and have a peek at how a guest OS must be modified to conform to achieve design goals of Xen . Xen was called as ***hypervisor***since it operated at a higher privilege level than the supervisor code of the guest operating systems that it hosted .

# 2.2 Hardware and Software Virtualization Overview .

The classical VMM implementations were based on ideas such as de-privileging, shadow structures and traces but creates obstacles to classical virtualization of the x86 architecture, as we know that Ignoring the legacy “real” and “virtual 8086” modes of x86, architected 32 and 64-bit protected modes are not virtualizable by classic principle . Main obstacles that’s on d way are *visibility of privileged state* and *Lack of traps when privileged instructions run at user-level .*Going Further into this paper we will see how binary translation (BT) overcomes the obstacles, and how adaptive BT takes efficiency up a notch after that , we will discuss architectural changes that permitted classical virtualization of the x86.VMware had implemented an experimental VMM to exploit those new hardware capabilities as hardware assisted VMM, and we shall compare it with the software VMM that was prevalent at that time .

# 2.3 Protean Overview

In this subsection we will have overview of Protean’s design. In Azure ,Demand is heterogeneous. Their zones exhibit workloads that are extremely diverse in operation and function . Team at Azure had observed that there are large number of different VM types and distribution of it is generally nonuniform .In most cases they had observed that most VMs require a small number of cores, but some require half or even an entire server. Scale and uncertainty analysis demonstrated that the incoming demand is highly variable. But Protean’s latency and throughput requirements cannot be compromised, and also Protean had to accommodate small and large regions .Evidence that subsequent requests are similar over time motivated them to put the “caching” of placement evaluation logic, and reuse across multiple requests .The Azure workload is highly diverse their sizes aren’t known in advance. This posed a substantial challenge in adequately packing the VM in physical servers.These are taken into account to develop Protean to work effectively as Rule based VM allocation service that scales across the azure platform .

# 3 Design of Components

# 3.1 Design of Xen

In this section we can have a look at the design of Xen-based server. We will see examples of Windows XP and NetBSD porting to show that Xen is indifferent to guest OSes .

There are 2 mechanisms that existed for control interactions between Xen and an overlying domain:

1.Hypercall : These are synchronous calls from a domain to Xen.

2.Event Mechanism : Xen Notifies underlying Domain via an asynchronous ***event*** mechanism.

Date transfer are handled via **I/O rings** , Two main factors have shaped the design of Xen’s I/O-transfer mechanism are resource management and event notification .Design of each ring is based around two pairs of producer consumer pointers: domains place requests on a ring, advancing a request producer pointer, and Xen removes these requests for handling, advancing an associated request consumer pointer. Responses are placed back on the ring .

Xen schedules domains according to the Borrowed Virtual Time (BVT) scheduling algorithm because it is work conserving and has a special mechanism for low-latency wake-up of a domain when it gets an event. Xen offers guest OSes with access to real time, virtual time and wall-clock time and Guest OSes are always expected to maintain internal timer queues and use the Xen-provided alarm timers to get timeout via async event mechanism .Xen tries to virtualize memory access with as little overhead as possible but x86 architecture’s use of hardware page tables makes it difficult to implement . In case of physical memory , it implements a *balloon driver ,* which adjusts a domain’s memory usage by passing memory pages back and forth between Xen and OS’s page allocator. And makes adjustments by using existing OS functions, to simplify functions . Out-of-memory handling mechanism in the guest OS can be changed as required automatically alleviate memory pressure.

# 3.2 Software vs Hardware VMs

3.2.1 Software VMM implementation

Software VMMs uses a Binary translator with properties given as follows :

Our software VMM uses a translator with these properties:

Input is binary x86 code,Translation happens at runtime, Code is translated only when it is about to execute and translator makes no assumptions about the guest code based on rules set by by x86 ISA with input being full x86 instruction se and output being safe subset then translated code is adjusted in response to guest behavior changes to improve overall efficiency. Simple BT eliminates traps from privileged instructions but frequent trap is still exists so we can use *adaptive BT*  to eliminate this kind of traps category of traps.

3.2.1 Hardware VMM implementation

The hardware extensions by hardware manufacturers provide a complete virtualization solution, giving us the structure of our hardware VMM .When running with a guest, the VMM fills in a VMCB with the existing guest state and executes *vmrun*. On guest exits, the VMM reads the VMCB fields to execute emulation code.

These emulation codes are shared with the software VMM that peripheral device models, code for interrupts and tasks such as logging, synch and interaction with the host OS. But current virtualzation hardware does not include explicit support for MMU virtualization so the hardware VMM takes help of software VMM’s implementation of the shadowing technique .

# 3.2 Protean VM resource allocatorImage

Above diagram gives general idea of protean’s architecture , we can have brief info on its design implementation .Cluster selection which is handled in milliseconds by filtering and sorting clusters instead of machines so Intuitively, the only computation that is required is to update the state by using caching approach . First we are caching internal rule state for efficient execution and then we update caching state with the latest changes. Usually they use Multiple rules to update same part of the state. So we have multiple rule based items to update cache system .

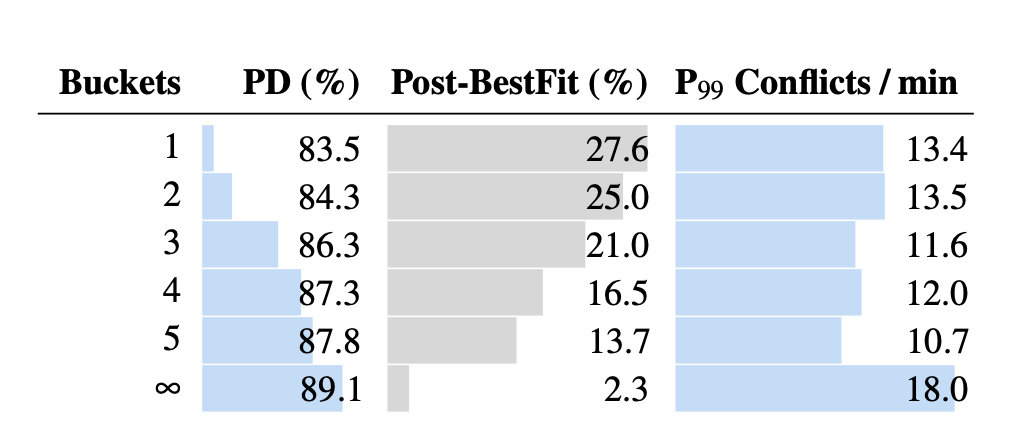
Apart from this these they often ran into conflicts when they ran huge clusters of machine when they handles with approaches such as fine grained conflict detection and handle it with by changing allocation quality that reduces conflict at end of its operation

# 4 Performances of various VMMs

# Xen’s Performance

Compared to native Linux , Xen offers only 3 percent overhead in CPU scheduling performance on virtualization , for network benchmarks With an MTU of 500 bytes, the per-packet overheads dominate. The extra complexity of transmit firewalling and receive demultiplexing adversely impact the throughput, but only by 14% . And also when we compare concurrency performance we get throughput scores for each domain are reflected in the different banding on each segment is within 4% of its expected size. For Perfomance isolation , we measured user-to-user UDP latency to one of the domains running the SPEC CINT2000 subset. To which we got mean response time of 5.4ms. even with significant back-ground load

# Protean’s Performance



Azure’s system handles the request volume currently seen in production and then they took an existing request trace and speed up time to1000x. They observed throughput reaches over 10k requests per second, but at expense of significant conflicts. As it eventually plateaus, likely due to a combination of fixed inventory size and increasing conflicts. They handled with numerous real time methods

# 6 Conclusion

I have presented a brief review on Virtualization papers that deals with Xen VMM , hardware visualization and Azure’s Protean VM allocater for large scale cloud computing .

All these papers deals with important advancement at different time period that acted as propellor upon which further improvements were made where all these aim to achieve only on things i.e to bring computing commodity computing resource to mainstream users. We are at the stage where computing resources are getting itself transformed to work similar to utility items . These advancements in last two decade has progressed us forward .

So What’s next ?

We will start seeing better VM allocation services that takes better risk and reward algorithms to bring further enhancements that’s both cost efficient and effective to both service providers and end users

**General References**

Xen and the Art of Virtualization by Paul Barham∗, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer†, Ian Pratt, Andrew Warfield

A Comparison of Software and Hardware Techniques for x86 Virtualization by Keith Adams and Ole Agesen

Protean: VM Allocation Service at Scale by Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda, *Microsoft Azure and Microsoft Research*