# Benchmarking Database Performance in a Virtual Environment

Sharada Bose, HP
sharada_bose@hp.com

Priti Mishra, Priya Sethuraman, Reza Taheri, VMWare, Inc.
{pmishra, psethuraman, rtaheri}@vmware.com

## Agenda/Topics

- Introduction to virtualization
- Performance experiments with benchmark derived from TPC-C
- Performance experiments with benchmark derived from TPC-E
- Case for a new TPC benchmark for virtual environments

# Variety of virtualization technologies

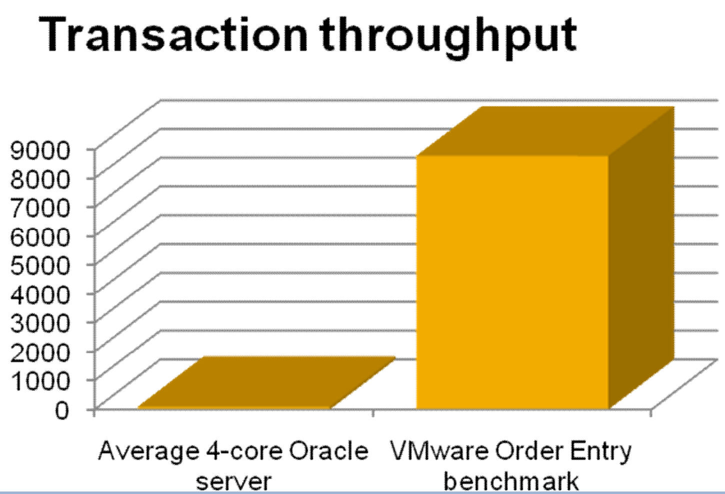
- IBM
  - System Z/VM and IBM PowerVM on the Power Systems
- Sun
  - X/VM and Zones
- HP
  - HP VM
- On the X86 processors
  - Xen and XenServer
  - Microsoft Hyper-V
  - KVM
  - VMware ESX
    - Oldest (2001) and largest market share
    - ***Where I work! So, focus of this talk***

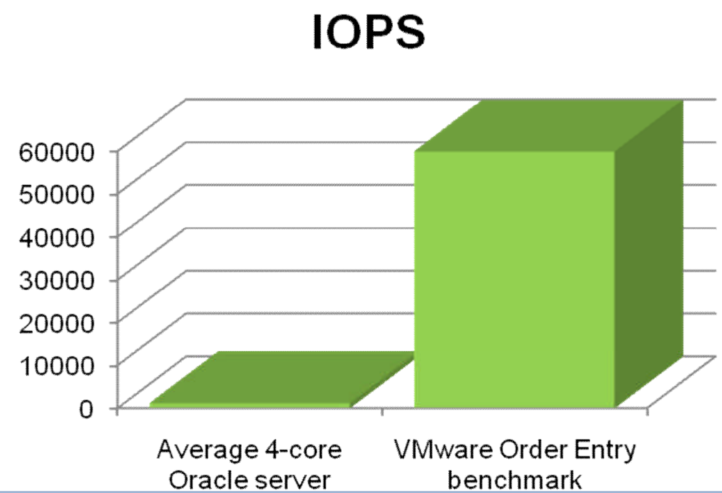
# Why virtualize?

- Server consolidation
  - The vast majority of server are grossly underutilized
  - Reduces both CapEx and OpEx
- Migration of VMs (both storage and CPU/memory)
  - Enables live load balancing
  - Facilitates maintenance
- High availability
  - Allows a small number of generic servers to back up all servers
- Fault tolerance
  - Lock-step execution of two VMs
- Cloud computing! Utility computing was finally enabled by
  - Ability to consolidate many VMs on a server
  - Ability to live migrate VMs in reaction to workload change

# How busy are typical servers?

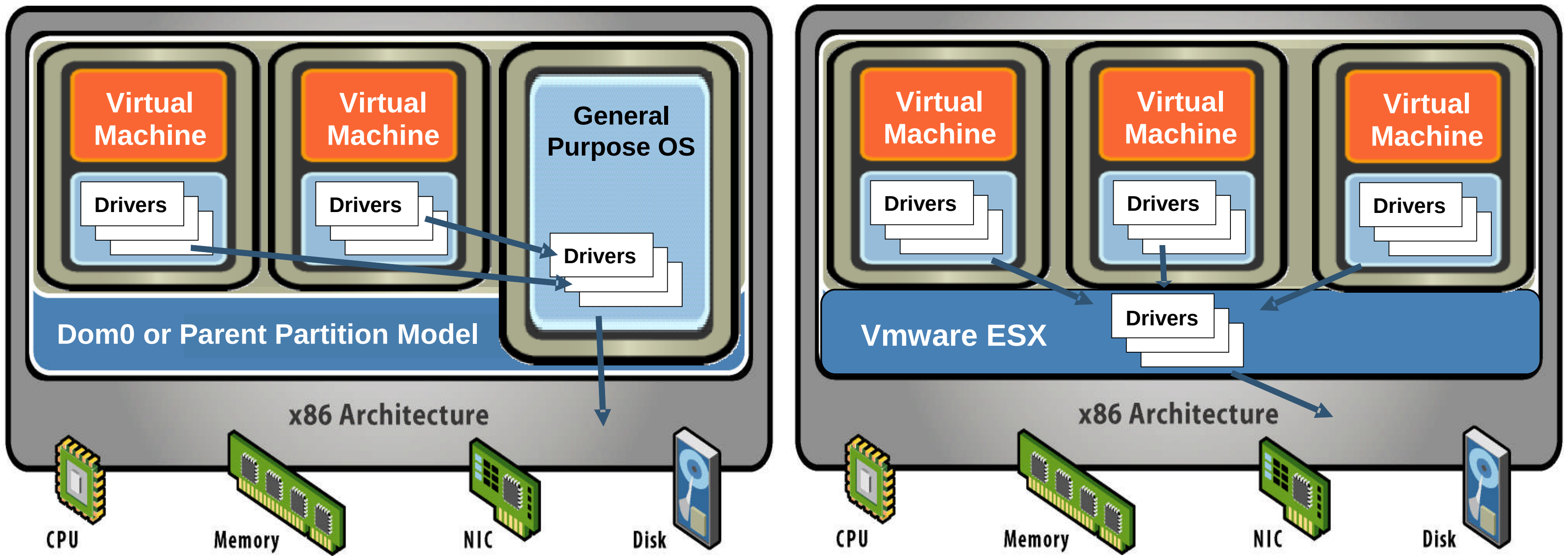
- Results of our experiment:
  - 8.8K DBMS transactions/second
  - 60K disk IOPS
- Typical Oracle 4-core installation:
  - 100 transactions/second
  - 1200 IOPSLock-step execution of two VMs




vmware

# Hypervisor Architectures




**Xen and Hyper-V**

- Very Small Hypervisor
- General purpose OS in parent partition for I/O and management
- All I/O driver traffic going thru parent OS

**ESX Server**

- Small Hypervisor < 24 mb
- Specialized Virtualization Kernel
- Direct driver model
- Management VMs
  - Remote CLI, CIM, VI API

vmware

# *Binary Translation* of Guest Code

**Translate guest kernel code**

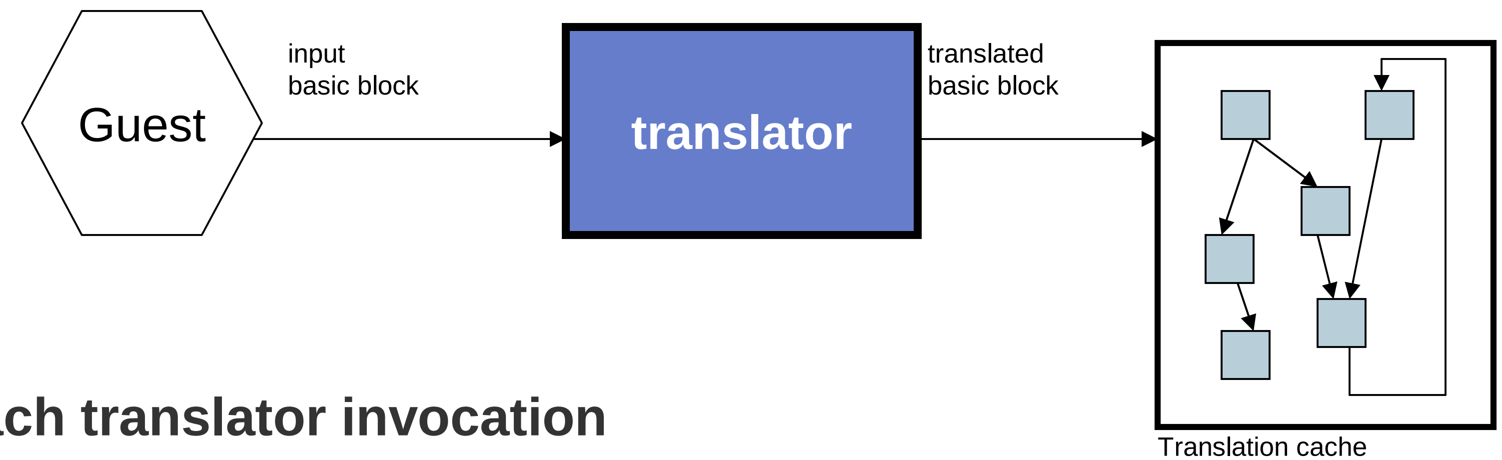
**Replace privileged instrs with safe “equivalent” instruction sequences**

**No need for traps**

**BT is an extremely powerful technology**

- Permits *any* unmodified x86 OS to run in a VM
- Can virtualize *any* instruction set

# BT Mechanics




## Each translator invocation

- Consume one input basic block (guest code)
- Produce one output basic block

## Store output in translation cache

- Future reuse
- Amortize translation costs
- Guest-transparent: no patching “in place”

# Virtualization Hardware Assist

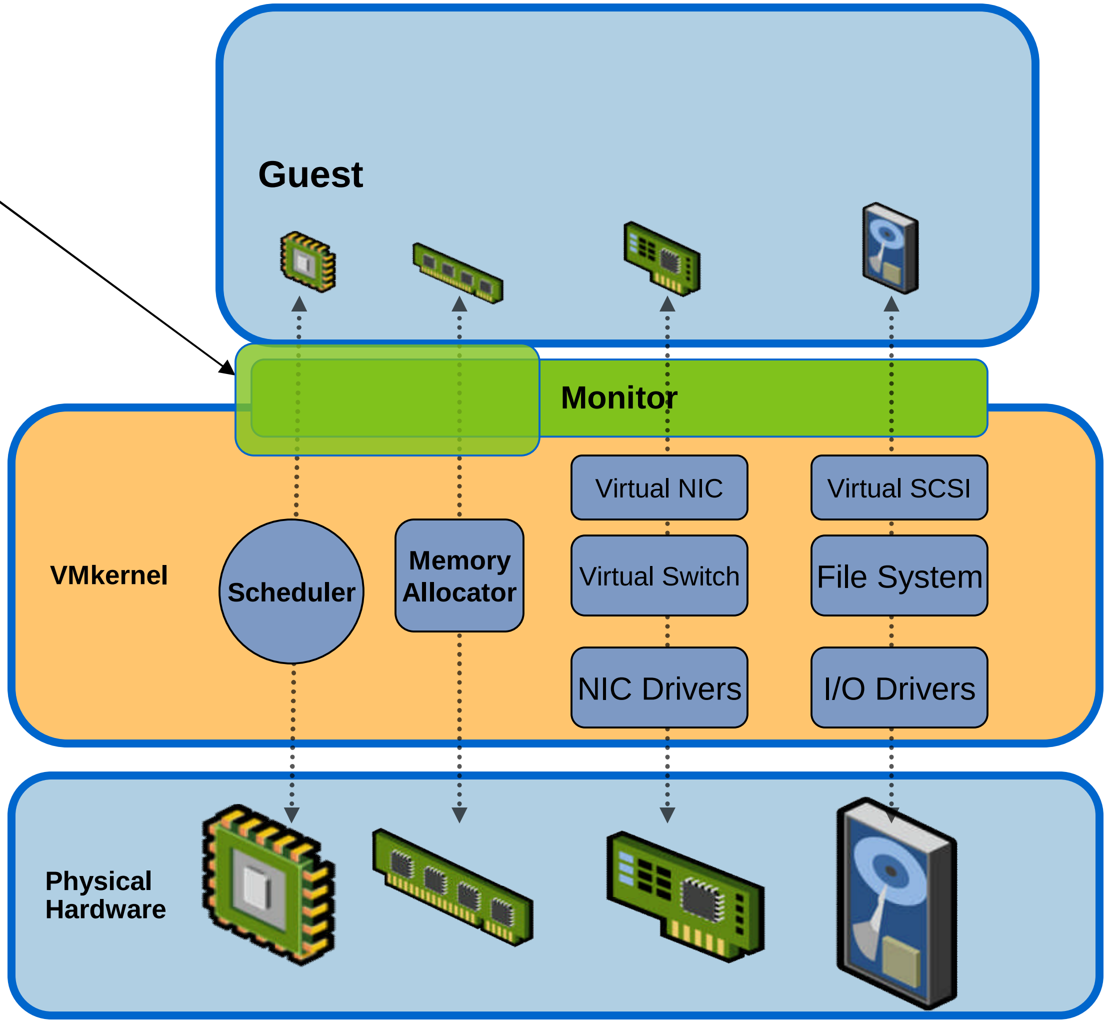
More recent CPUs have features to reduce some of the overhead at the monitor level

Examples are Intel VT and AMD-V

Hardware-assist doesn't remove all virtualization overheads: scheduling, memory management and I/O are still virtualized with a software layer

The Binary Translation monitor is faster than hardware-assist for many workloads

VMware ESX takes advantage of these features.




vmware

# Performance of a VT-x/AMD-V Based VMM

**VMM only intervenes to handle exits**

**Same performance equation as classical trap-and-emulate:**

- overhead = exit frequency * average exit cost

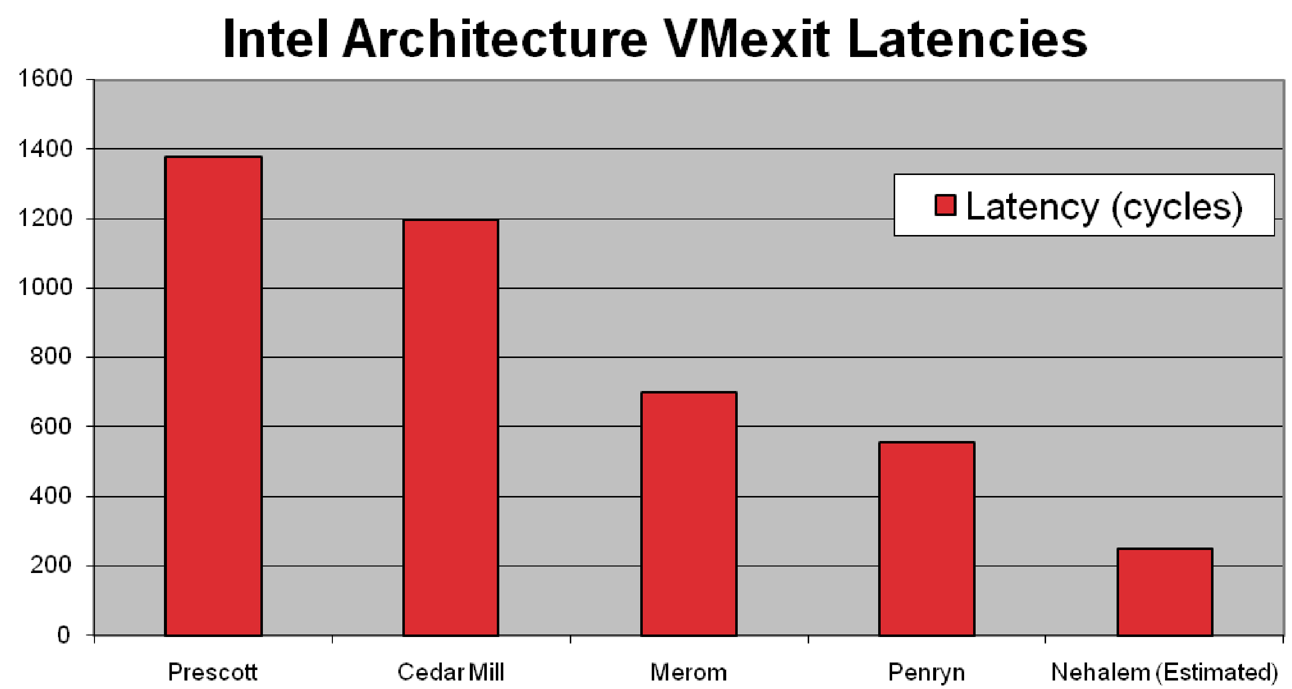
**VMCB/VMCS can avoid simple exits (e.g., enable/disable interrupts), but many exits remain**

- Page table updates
- Context switches
- In/out
- Interrupts

# Qualitative Comparison of BT and VT-x/AMD-V

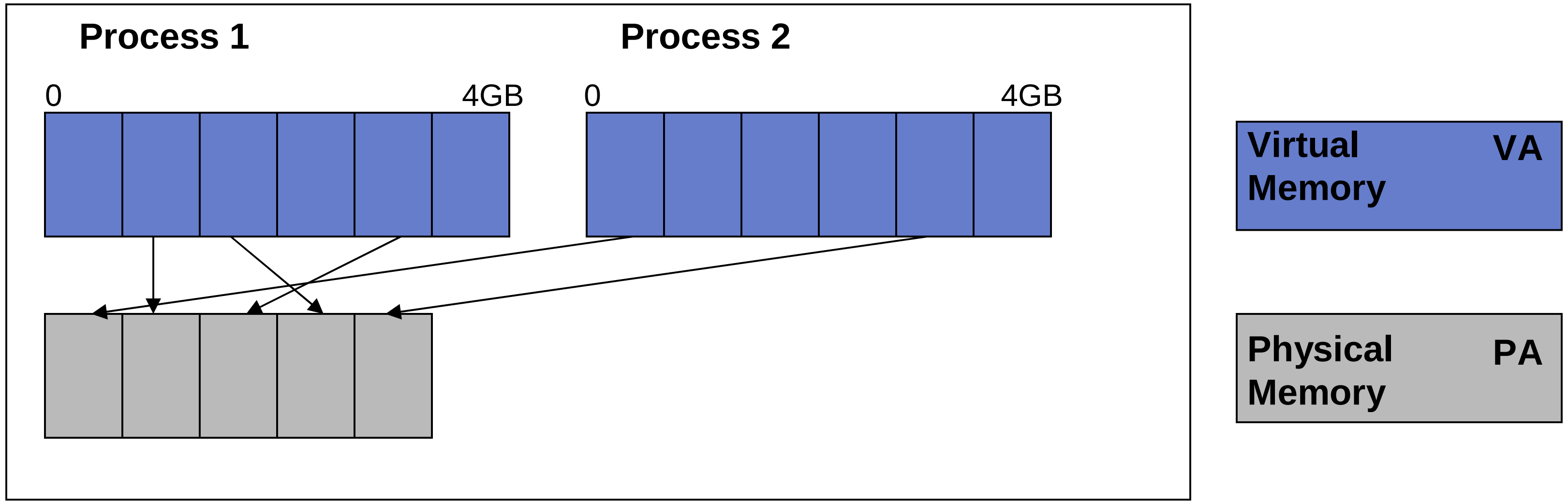
- **BT loses on:**
  - system calls
  - translator overheads
  - path lengthening
  - indirect control flow
- **BT wins on:**
  - page table updates (adaptation)
  - memory-mapped I/O (adapt.)
  - IN/OUT instructions
  - no traps for priv. instructions
- **VT-x/AMD-V loses on:**
  - exits (costlier than “callouts”)
  - no adaptation (cannot elim. exits)
  - page table updates
  - memory-mapped I/O
  - IN/OUT instructions
- **VT-x/AMD-V wins on:**
  - system calls
  - almost all code runs “directly”

# VMexit Latencies are getting lower…

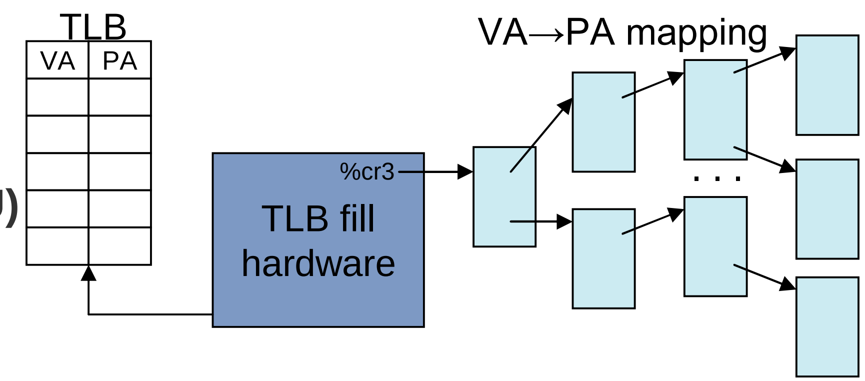



- VMexit performance is critical to hardware assist-based virtualization
- In additional to generational performance improvements, Intel is improving VMexit latencies

# Virtual Memory (ctd)

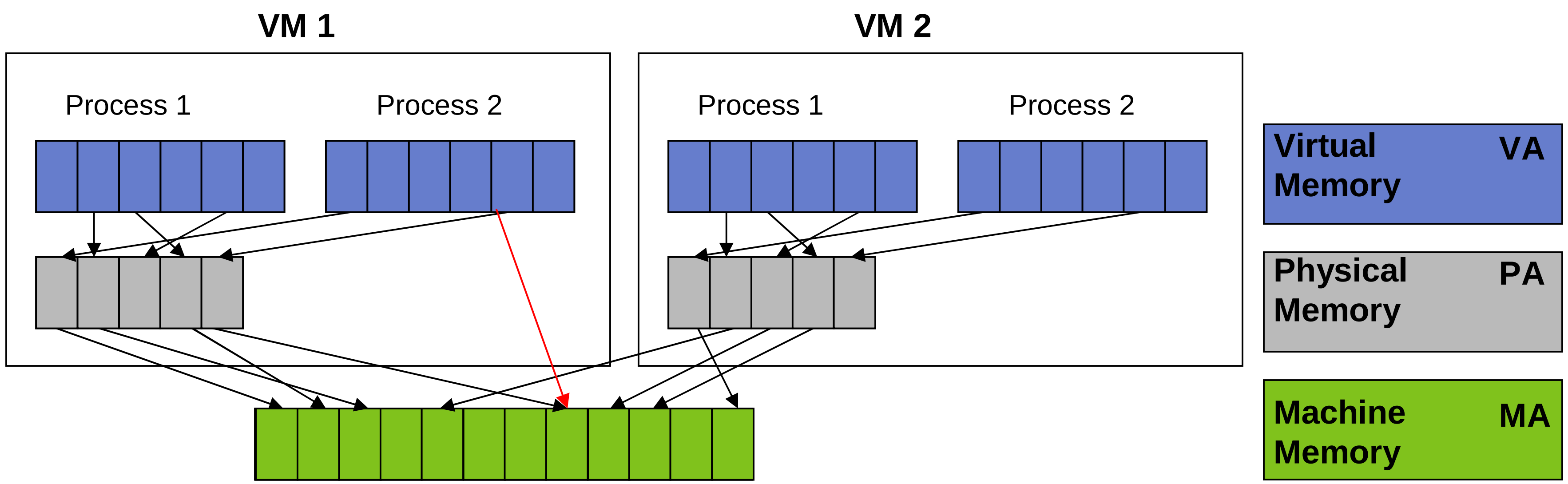



- Applications see contiguous virtual address space, not physical memory
- OS defines VA -> PA mapping
  - Usually at 4 KB granularity
  - Mappings are stored in page tables
- HW memory management unit (MMU)
  - Page table walker
  - TLB (translation look-aside buffer)



# Virtualizing Virtual Memory

*Shadow Page Tables*

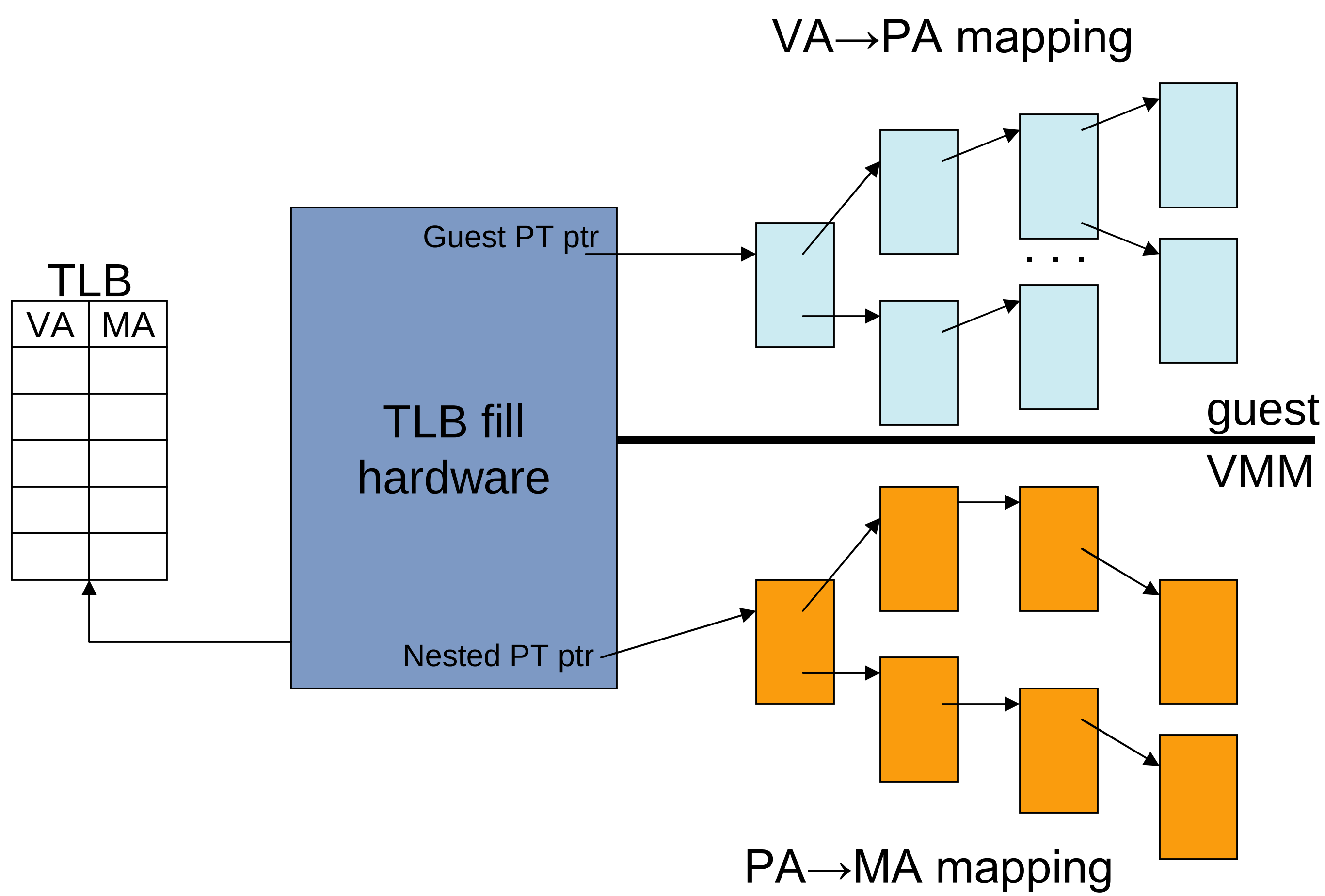



**VMM builds "shadow page tables" to accelerate the mappings**

- Shadow directly maps VA -> MA
- Can avoid doing two levels of translation on every access
- TLB caches VA->MA mapping
- Leverage hardware walker for TLB fills (walking shadows)
- When guest changes VA -> PA, the VMM updates shadow page tables

# $2^{nd}$ Generation Hardware Assist

*Nested/Extended Page Tables*



# Analysis of NPT

**MMU composes VA->PA and PA->MA mappings *on the fly* at TLB fill time**

## Benefits

- Significant reduction in "exit frequency"
  - No trace faults (primary page table modifications as fast as native)
  - Page faults require no exits
  - Context switches require no exits
- No shadow page table memory overhead
- Better scalability to wider vSMP
  - Aligns with multi-core: performance through parallelism

## Costs

- More expensive TLB misses: $O(n^2)$ cost for page table walk, where n is the depth of the page table tree

# CPU and Memory Paravirtualization

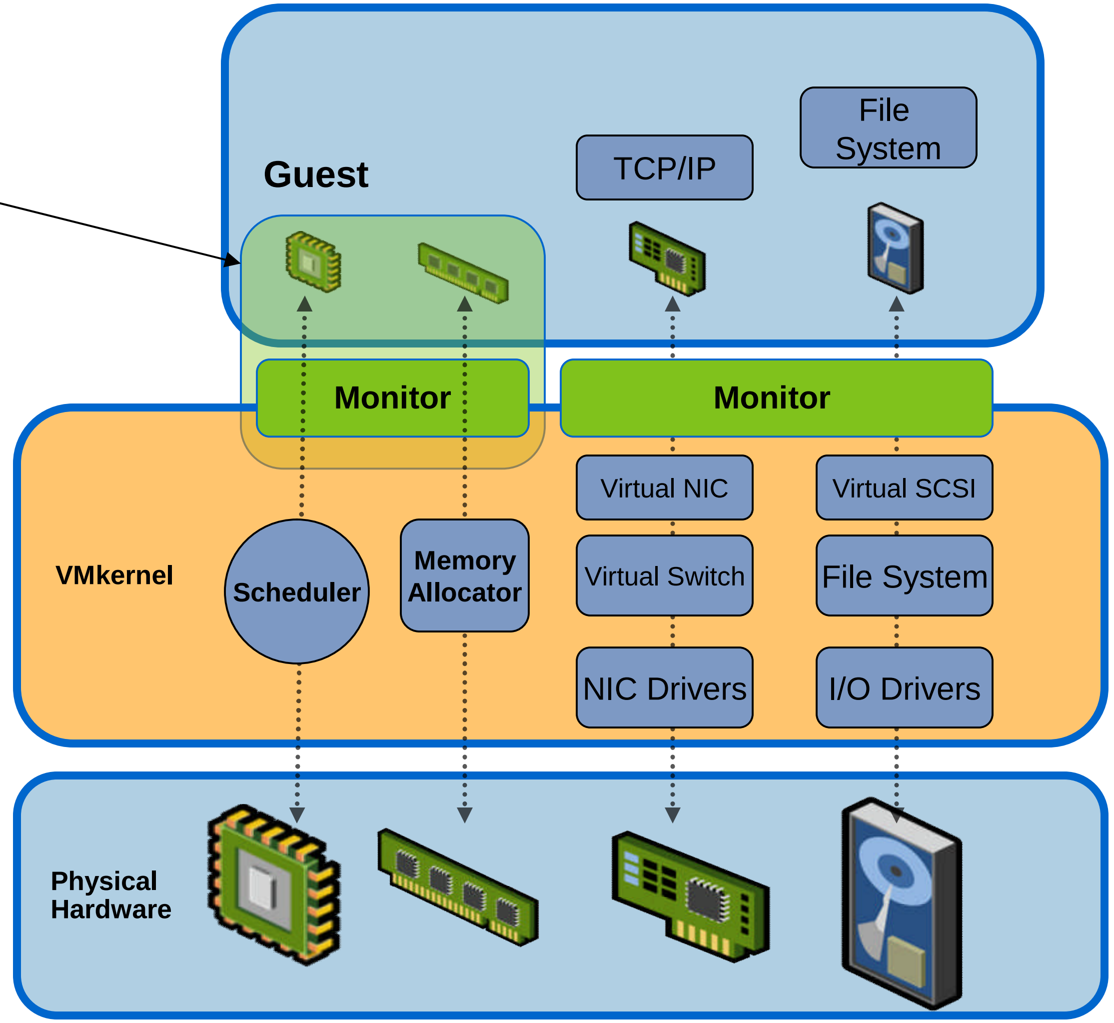
Paravirtualization extends the guest to allow direct interaction with the underlying hypervisor

Paravirtualization reduces the monitor cost including memory and System call operations.

Gains from paravirtualization are workload specific

Hardware virtualization mitigates the need for some of the paravirtualization calls

VMware approach:
VMI and paravirt-ops




vmware

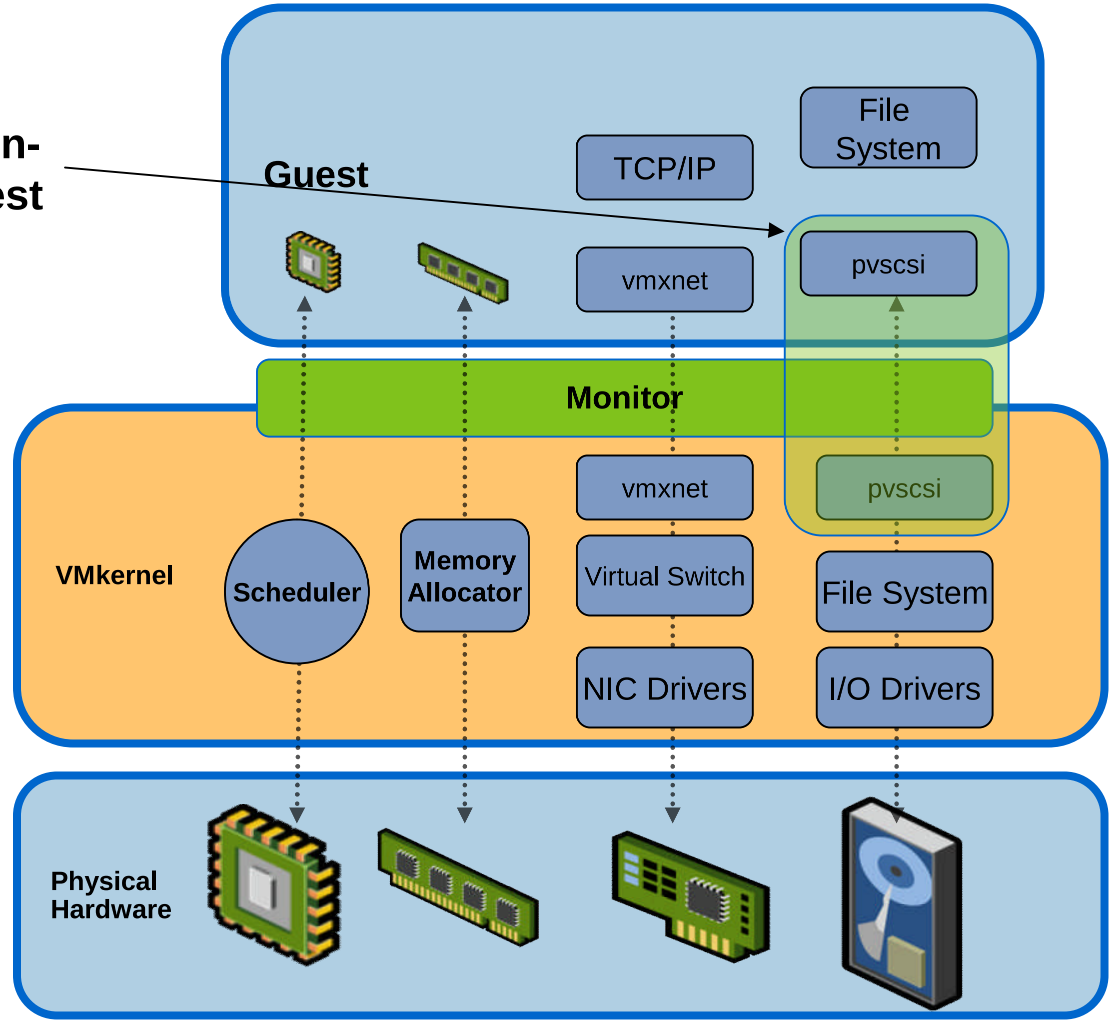
# Device Paravirtualization

**Device Paravirtualization places A high performance virtualization-Aware device driver into the guest**

**Paravirtualized drivers are more CPU efficient (less CPU over-head for virtualization)**

**Paravirtualized drivers can also take advantage of HW features, like partial offload (checksum, large-segment)**

**VMware ESX uses para-virtualized network and storage drivers**



# Paravirtualization

- For performance
  - Almost everyone uses a paravirt driver for mouse/keyboard/screen and networking
  - For high throughput devices, makes a big difference in performance
- Enabler
  - Without Binary Translation, the only choice on old processors
    - Xen with Linux guests
  - Not needed with newer processors
    - Xen with Windows guests

# Today's visualization benchmarks

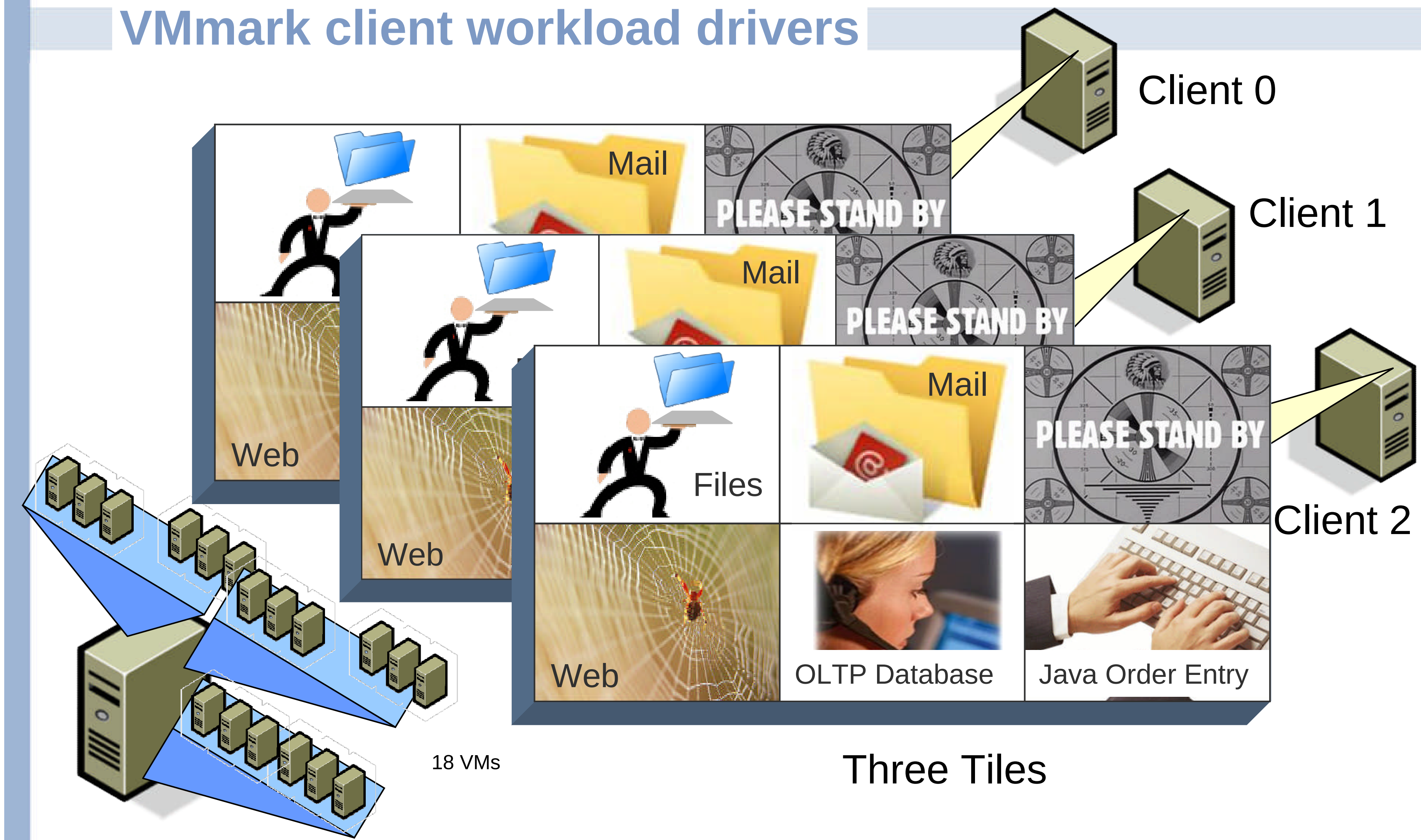
- VMmark
  - Developed by VMware in 2007
  - De facto industry standard
  - 84 results from 11 vendors
- SPECvirt
  - Still in development
  - Will likely become ***the*** virtualization benchmark
  - But not a DBMS/backend server benchmark
- vConsolidate
  - Developed by IBM and Intel in 2007
- vApus Mark I from Sizing Server Lab
- vServCon developed for internal use by Fujitsu Siemens Computers

# VMmark

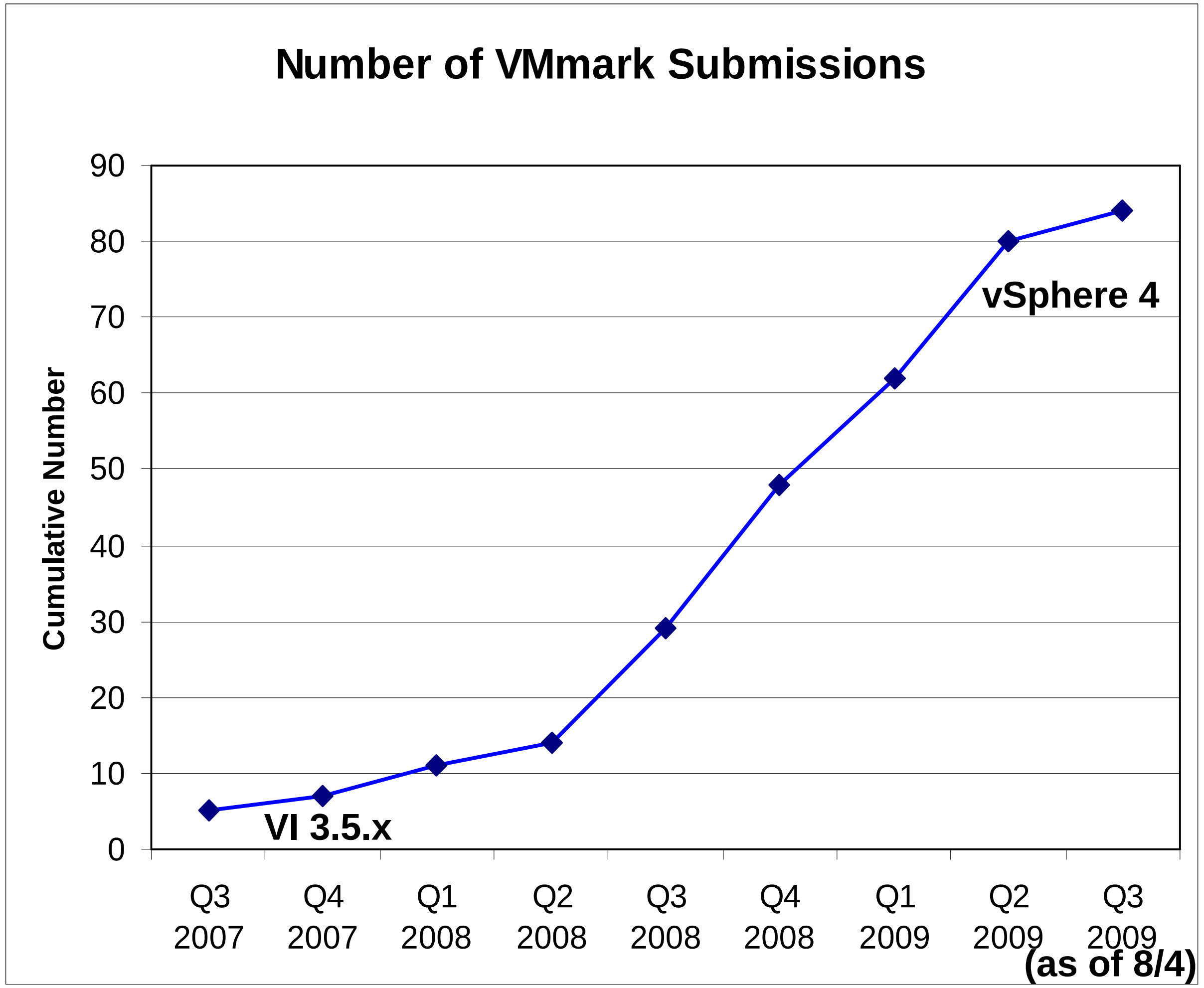
- Aimed at server consolidation market
- *A mix of workloads*
  - ***Tile*** is a collection of VMs executing a set of diverse workloads

| Workload | Application | Virtual Machine Platform |
|---|---|---|
| Mail server | Exchange 2003 | Windows 2003, 2 CPU, 1GB RAM, 24GB disk |
| Java server | SPECjbb®2005-based | Windows 2003, 2 CPU, 1GB RAM, 8GB disk |
| Standby server | None | Windows 2003,1 CPU, 256MB RAM, 4GB disk |
| Web server | SPECweb®2005-based | SLES 10, 2 CPU, 512MB RAM, 8GB disk |
| Database server | MySQL | SLES 10, 2 CPU, 2GB RAM, 10GB disk |
| File server | dbench | SLES 10, 1 CPU, 256MB RAM, 8GB disk |

# VMmark client workload drivers



# VMmark is the de-facto Virtualization Benchmark



# So why do we need a new benchmark?

- Most virtual benchmarks today cover consolidation of ***diverse workloads***
- None are aimed at transaction processing or decision support applications, the traditional areas addressed by TPC benchmarks.
- The new frontier is virtualization of resource-intensive workloads, including those which are distributed across multiple physical servers.
- None of the existing virtual benchmarks available today measure the database-centric properties that have made TPC benchmarks the industry standard that they are today.

## But is virtualization ready for a TPC benchmark?

- The accepted industry lore has been that databases are not good candidates for virtualization
- In the following slides, we will show that benchmarks derived from TPC workloads run extremely well in virtual machines
- We will show that there exists a natural extension of existing TPC benchmarks into new virtual versions of the benchmarks

# Databases: Why Use VMs for databases?

**Virtualization at hypervisor level provides the best abstraction**

- Each DBA has their own hardened, isolated, managed sandbox

**Strong Isolation**

- Security
- Performance/Resources
- Configuration
- Fault Isolation

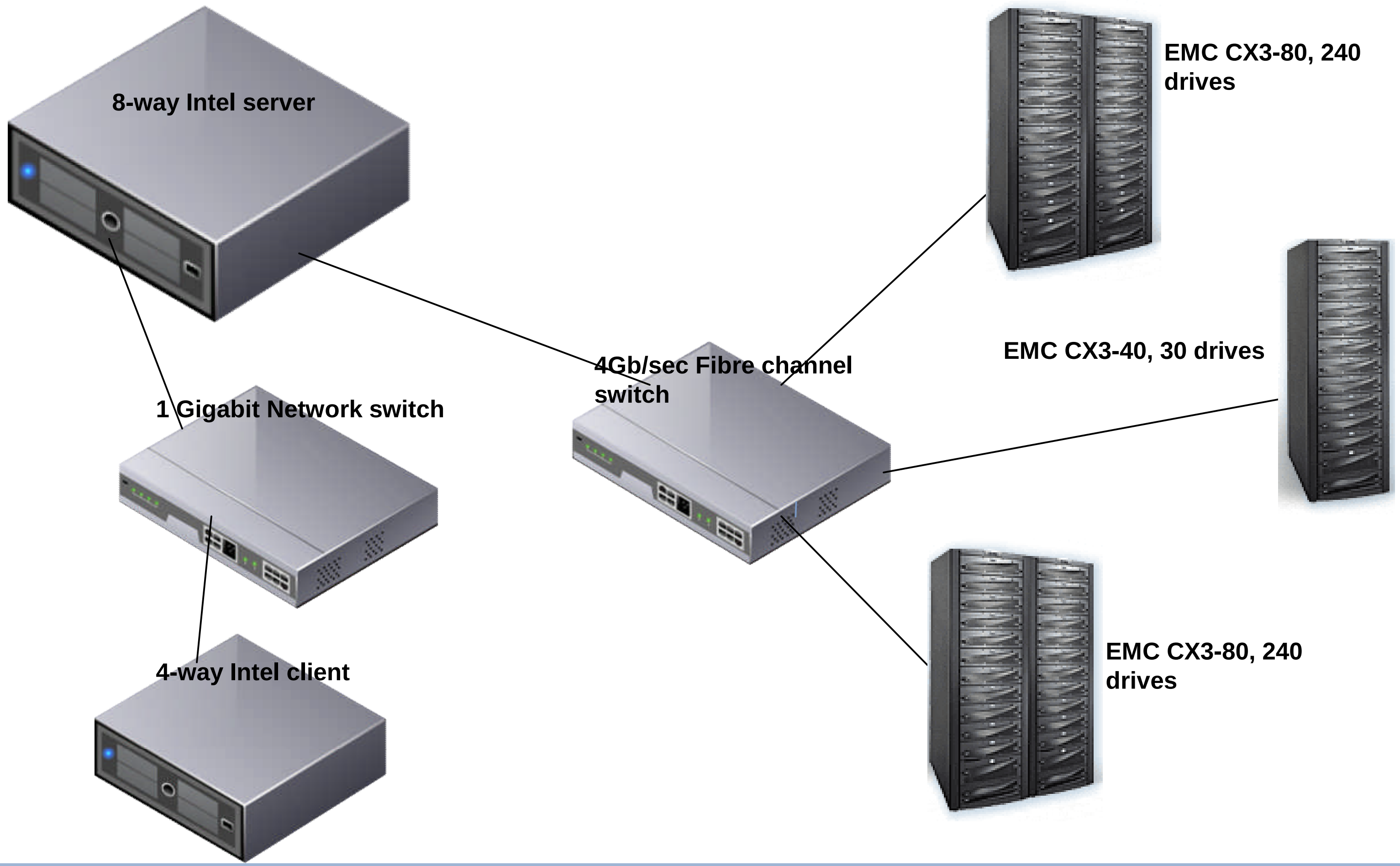
**Scalable Performance**

- Low-overhead virtual Database performance
- Efficiently Stack Databases per-host

# First benchmarking experiment

- Workload: Pick a workload that is:
  - A database workload
  - OLTP
  - Heavy duty
  - A workload that everybody knows and understands
  - So we decided on a benchmark that is a fair-use implementation of the TPC-C business model
    - ***Not compliant TPC-C results. Results cannot be compared to official TPC-C publications***

# Configuration, Hardware



## Configuration, Benchmark

- The workload is borrowed from the TPC-C benchmark; let us call this the Order Entry Benchmark
- A *batch* benchmark; there were up to 625 DBMS client processes running on a separate client computer, generating the load
- 7500 warehouses and a 28GB SGA
  - We were limited by the memory available to us; hence a DB size smaller than the size required for our throughput. With denser DIMMs, we would have used a larger SGA and a larger database
  - Our DBMS size/SGA size combination puts the same load on the system as ~17,000 warehouses on a 72GB-system
  - Reasonable database size for the performance levels we are seeing

## Disclaimers

**ACHTUNG!!!**

- All data is based on in-lab results w/ a developmental version of ESX
- Our benchmarks were fair-use implementations of the TPC-C and TPC-E business models; our results are not TPC-C|E compliant results, and not comparable to official TPC-C|E results. TPC Benchmark is a trademark of the TPC.
- Our throughput is not meant to indicate the absolute performance of Oracle and MS SQL Server, or to compare their performance to another DBMSs. Oracle and MS SQL Server were simply used to analyze a virtual environment under a DBMS workload
- Our goal was to show the relative-to-native performance of VMs, and the ability to handle a heavy database workload, not to measure the absolute performance of the hardware and software components used in the study

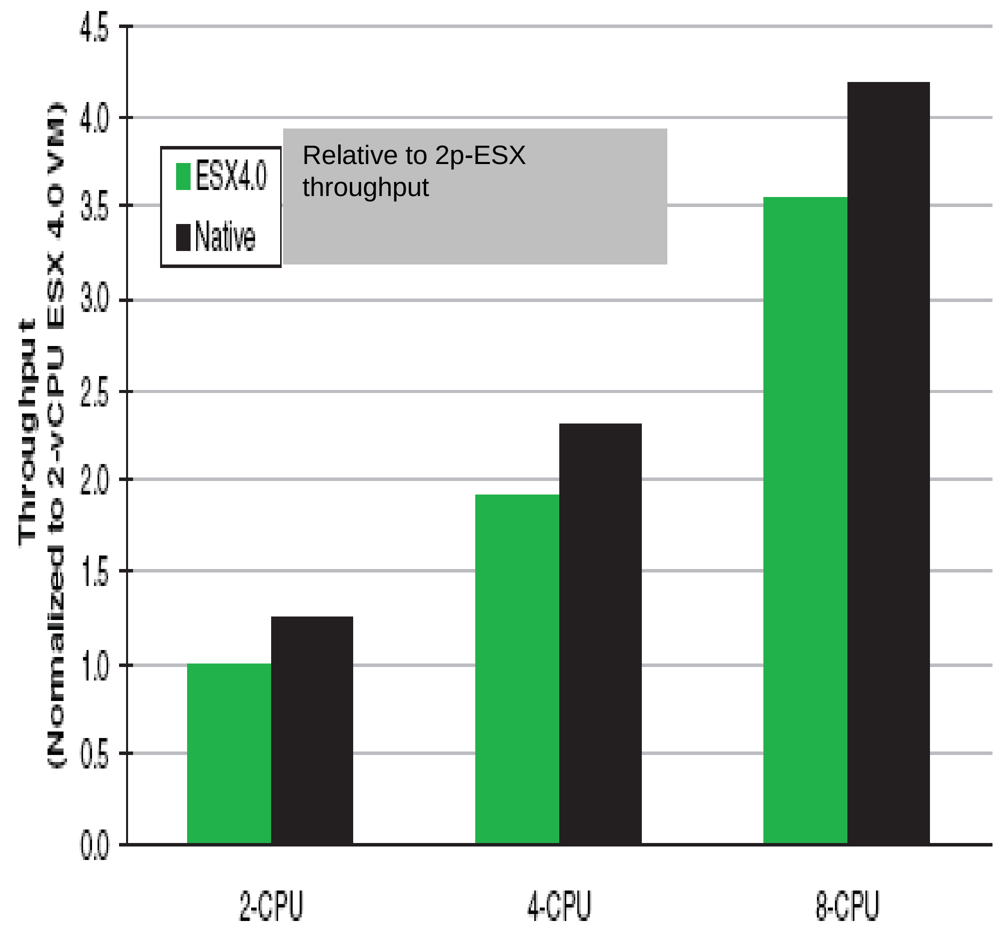
# Results: Peak

- The VM throughput was 85% of native throughput
- Impressive in light of the heavy kernel mode content of the benchmark
- Results summary for the 8-vcpu VM:

| Configuration | Native | VM |
|---|---|---|
| **Throughput in business transactions per minute** | **293K** | **250K** |
| **Disk IOPS** | **71K** | **60K** |
| **Disk Megabytes/second** | **305 MB/s** | **258 MB/s** |
| **Network packets/second** | **12K/s receive**<br>**19K/s send** | **10K/s receive**<br>**17K/s send** |
| **Network bandwidth/second** | **25Mb/s receive**<br>**66Mb/s send** | **21Mb/s receive**<br>**56Mb/s send** |

## Results:
## ESX4.0 vs. Native Scaling

- VM configured with 1, 2, 4, and 8 vCPUs
- In each case, ESX was configured to use the same number of pCPUs
- Each doubling of vCPUs results in ~1.9X increase in throughput



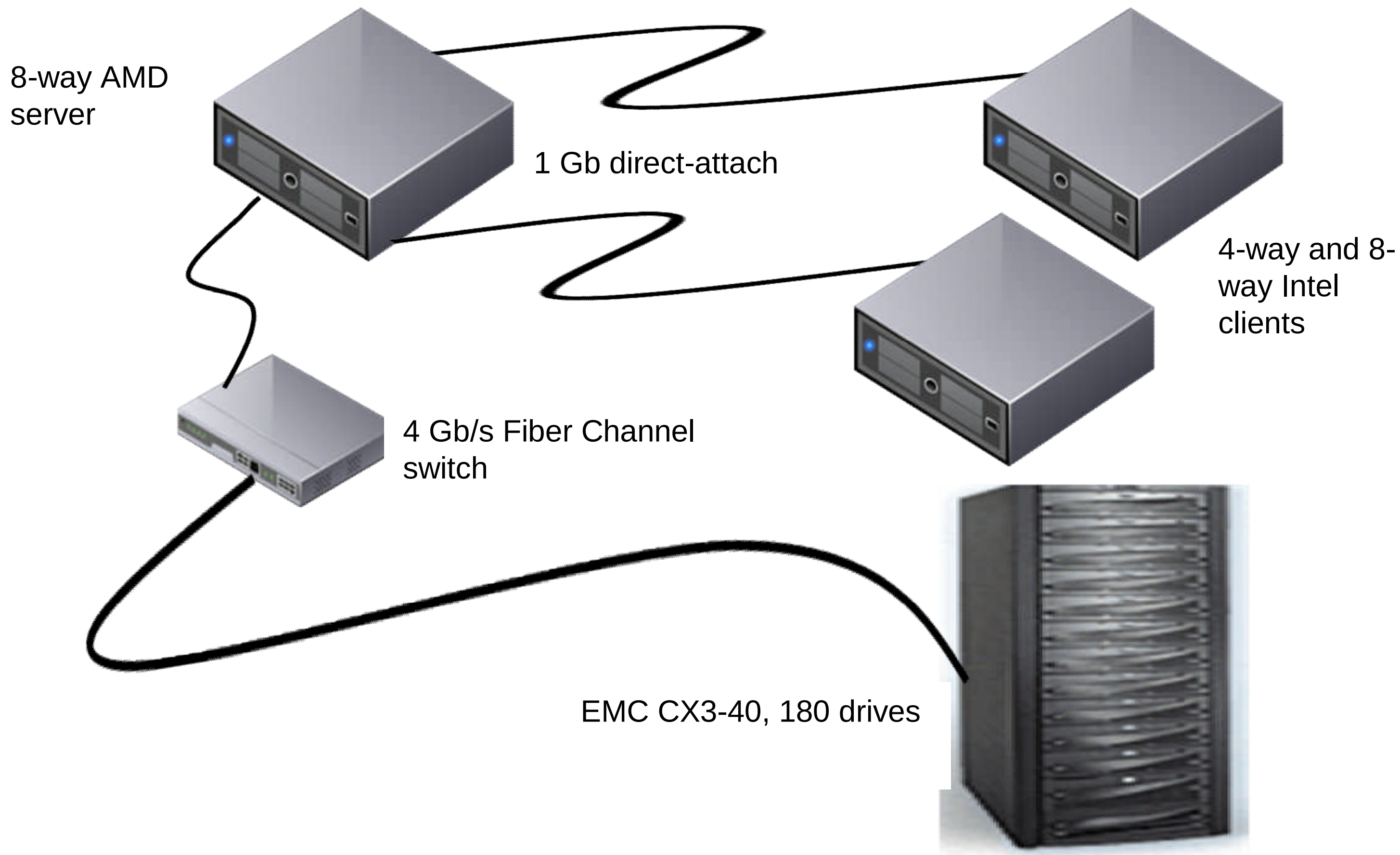
# SQLServer Performance Characteristics

- **Non-comparable implementation of TPC-E**
  - Models a brokerage house
  - Complex mix of heavyweight transactions

| Metric | 4VCPU VM |
|---|---|
| Database size | 500 GB |
| Disk IOPS | 10500 |
| SQLServer buffer cache | 52 GB |
| Network Packets/sec | 7,500 |
| Network Throughput | 50 Mb/s |

vmware

# Hardware configuration for tests on vSphere 4.0

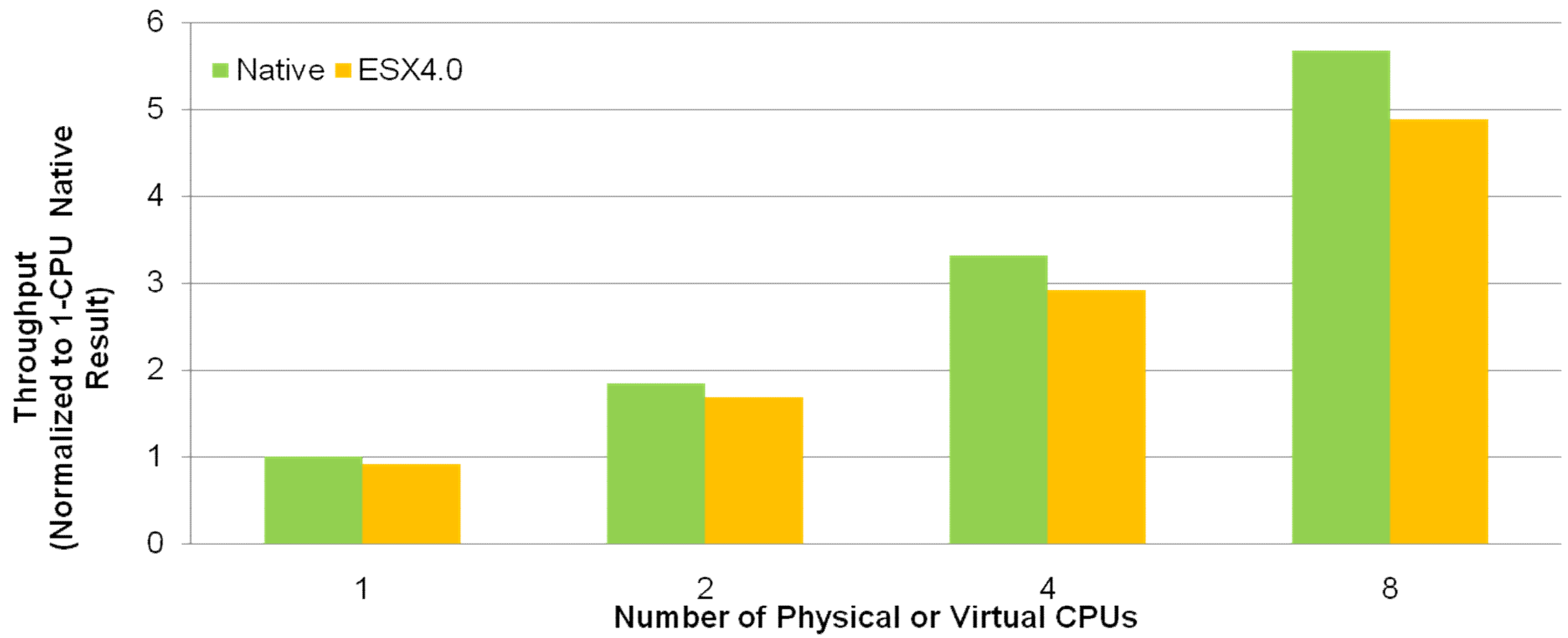


# Resource intensive nature of the 8-vCPU VM

| **Metric** | **Physical Machine** | **Virtual Machine** |
|---|---|---|
| Throughput in transactions per second* | 3557 | 3060 |
| Average response time of all transactions** | 234 milliseconds | 255 milliseconds |
| Disk I/O throughput (IOPS) | 29 K | 25.5 K |
| Disk I/O latencies | 9 milliseconds | 8 milliseconds |
| Network packet rate receive<br><br>Network packet rate send | 10 K/s<br><br>16 K/s | 8.5 K/s<br><br>8 K/s |
| Network bandwidth receive<br><br>Network bandwidth send | 11.8 Mb/s<br><br>123 Mb/s | 10 Mb/s<br><br>105 Mb/s send |

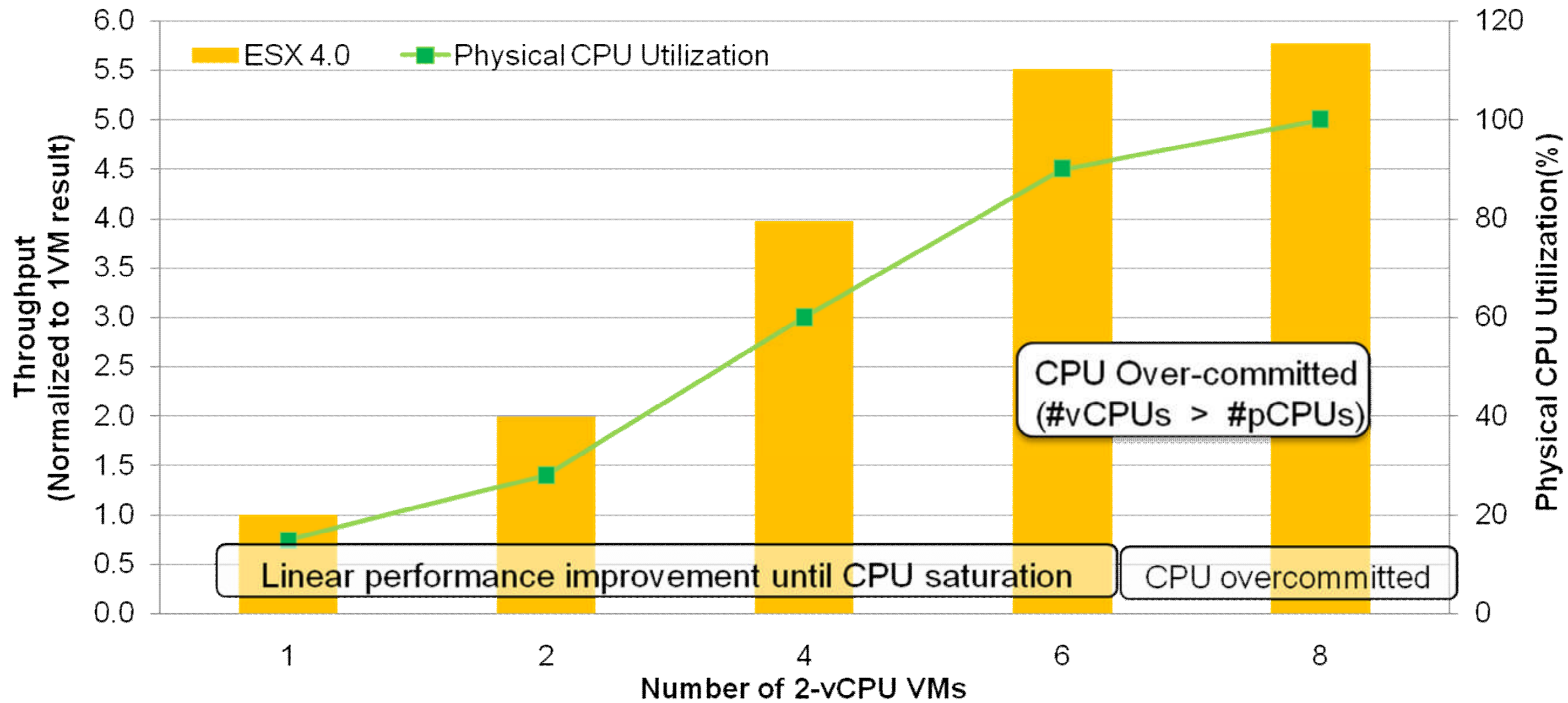
vmware

# SQL Server Scale up performance relative to native



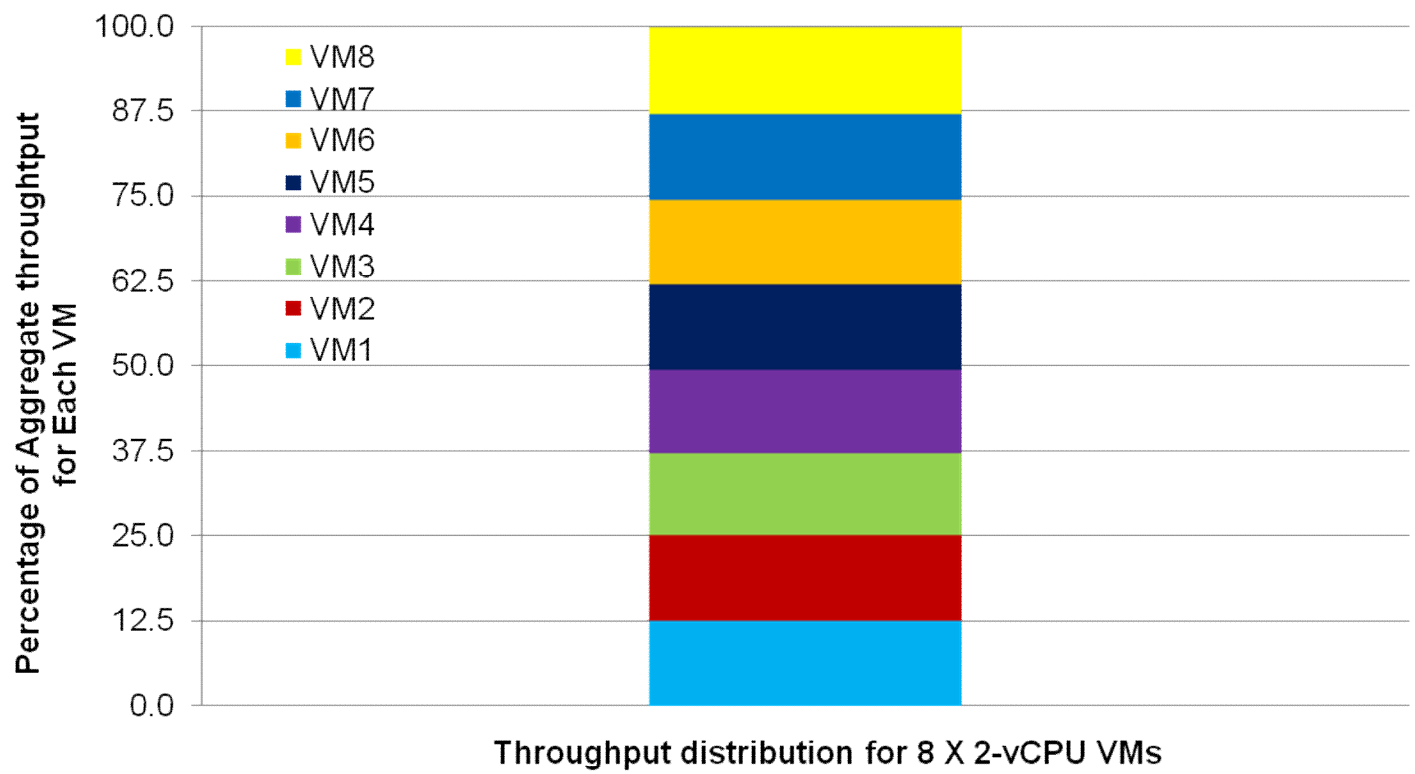

- At 1 & 2 vCPUs, ESX is 92 % of native performance
  - Hypervisor able to effectively offload certain tasks to idle cores.
  - flexibility in making virtual CPU scheduling decisions
- 4 vCPUs , 88% and 8 vCPUs 86 % of native performance

# SQL Server Scale out experiments




- Throughput increases linearly as we add up to 8vCPUs in four VMs
- Over-committed, going from 4 to 6 VMs (1.5x), performance rises 1.4x

# Scale out overcommittment fairness




- Fair distribution of resources to all eight VMs

## Benchmarking databases in virtual environments

We have shown database are good candidates for virtualization

- But no formal benchmark
- Can benchmark a single VM on the server
  - IBM's power series TPC disclosures
- Need a TPC benchmark to cover the multi-VM case

***It is what the users are demanding!***

# Proposal 1

Comprehensive database virtualization benchmark

- Virtual machine Configuration:
  - System should contain a mix of at least two multi-way CPU configurations, for example an 8-way server result might contain 2x2 vCPU and 1x4 vCPU VMs
  - Measure the cpu overcommitment capabilities in hypervisors by providing an overcommitted result along with a fully committed result.
  - Both results should report throughput of individual VMs.
- Workloads used
  - Each VM runs homogenous or heterogeneous workloads of a mix of database benchmarks, e.g., TPC-C, TPC-H and TPC-E.
  - Consider running a mix of operating systems and databases.

# Proposal 1

- Advantages
  - Comprehensive database consolidation benchmark
- Disadvantages
  - Complex benchmark rules may be  too feature-rich for an industry standard workload

## Proposal 2

Virtualization extension of an existing database benchmark

- Virtual Machine configuration:
  - System contains a mix of homogenous VMs, for example an 8-way server might contain 4x2 vCPU VMs
  - The number of vCPUs in a VM would be based on the total number of cores and the cores/socket on a given host
    - E.g., an 8-core has to be 4 2-vCPU VMs; a 64-core 8 8-vCPU VMs
  - The benchmark specification would prescribe the number of VMs and number of vCPUs in each VM for a given number of cores
- Workloads used
  - Homogeneous database workload, e.g., TPC-E, in each VM

# Proposal 2

- Advantages
  - Simple approach provides users with a wealth of information about virtualized environments that they do not have currently
  - The simplicity of the extension makes it possible to develop a new benchmark quickly, which is critical if the benchmark is to gain acceptance
- Disadvantages
  - Unlike Scenario 1, this approach does not emualte consolidation of diverse workloads
  - Features of virtual environments such as over-commitment not part of the benchmark definition

# Proposal 3

Benchmarking multi-tier/multi-phase applications

- map each step in a workflow (or, each tier in a multi-tier application) to a VM. (For large-scale implementations, mapping may instead be to a set of identical/homogeneous VMs.)
- From a benchmark design perspective, a challenging exercise with a number of open questions, e.g.:
  - Does the benchmark specify strict boundaries between the tiers?
  - Are the size and number of VMs in each layer parts of the benchmark spec?
  - Does the entire application have to be virtualized? Or, would benchmark sponsors have freedom in choosing the components that are virtualized? This question arises due to the fact that support and licensing restrictions often lead to parts not being virtualized.

# Recommendation

- TPC benchmarks are great, but take a long time to develop
  - Usually well worth the wait
  - But in this case, timing is everything
- So, go for something simple: an extension of an existing benchmark
- Proposal #2 fits the bill
  - Not esoteric, is what most users want
  - Can be developed quickly
  - Based on a proven benchmark
  - Yes, it is really that simple!

# Conclusions

- Virtualization is a mature technology in heavy use by customers
- Databases were the last frontier; we have shown it's been conquered
- Benchmarking community is behind the curve
- Badly in need of a TPC benchmark
- A simple extension of TPC-E is:
  - A natural fit
  - Easy to produce
  - Timely
  - Great price performance!