EC 440 – Introduction to Operating Systems

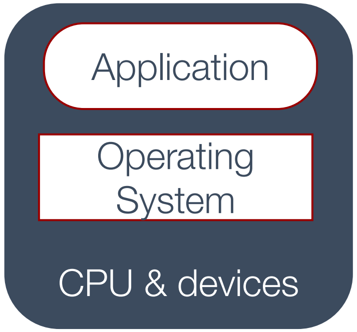
Orran Krieger (BU)
Larry Woodman (Red Hat)

Virtualization

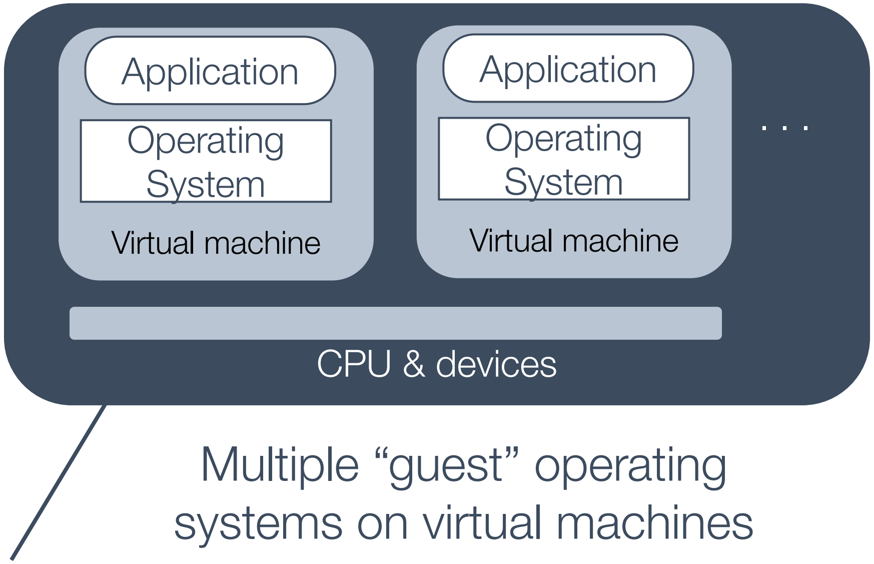
What is it?

- A software-implemented *virtual machine/computer* that you can run a full OS on top of
- Started way back in mid 60s, share expensive computer between multiple single user Oses
- Went away (outside of mainframe) when OSes got better….
- Then… came back in the 90s with VMware

Simple view



“Bare Metal”



Multiple “guest” operating systems on virtual machines

“hypervisor” or “virtual machine monitor”

My story in the late 90s

- Demonstrates limitations in current OSes:
  - Scalable apps on non-scalable OSes (Disco) & Fault containment (Cellular Disco)
    - Hive-> Disco (Stanford/Mendel)
    - Hurricane -> Tornado/K42 (Toronto & IBM)
  - Server consolidation because the OSes can't isolate the workloads – windows sucks
- Keeps the OS from direct access to the HW.
- Large grained partitioning of resources result in inefficiencies.
- Makes fine grained sharing difficult.
- Requires configuration and management of multiple OSes.

Hypervisors are for weenies

Then Sony came along…

- Game comes with its own customized OS.
- Absolutely deterministic performance.
  - OS can *never* be upgraded.
- But…. Need persistent storage, network access, general applications…
- Solution: Hypervisor with General purpose Linux and ability to start games in their own domain/partition.
  - Talked to product team; 100 person year
  - We built IBM's "Research Hypervisor" (rHype) in 2 using paravirtualization
- Then…. got the religion
  - IBM's "Research Hypervisor" (rHype).
  - hypervisor for HPC/PERCS
  - architect Xen (PPC)
  - Adjunct partition PHYP – Linux DD

What are the advantages virtualization?

My story in 2005 (confidential)

- Value proposition extends beyond high end servers:
  - On-demand
  - RAS
  - Client-server security
  - Windows monopoly
  - ISV testing and certification
  - OS development
  - Real time
  - Silicon IP
  - HPC OSes
  - Next-gen Game console
  - Migration/checkpoint/restart…
  - HW upgrading
  - Architecture evolution
  - OEM differentiation
  - Grid
  - …

- Virtualization will become commodity and ubiquitous: VMWare, Microsoft Virtual PC & Virtual Server, UML, coLinux, Xen, Denali, L4, Jaluna…
- We can lead this, opportunities for PCD, Tivoli, pixSeries, SW…
- Result – "only Research will ever think of this…" … "will kill IBM's server brands"

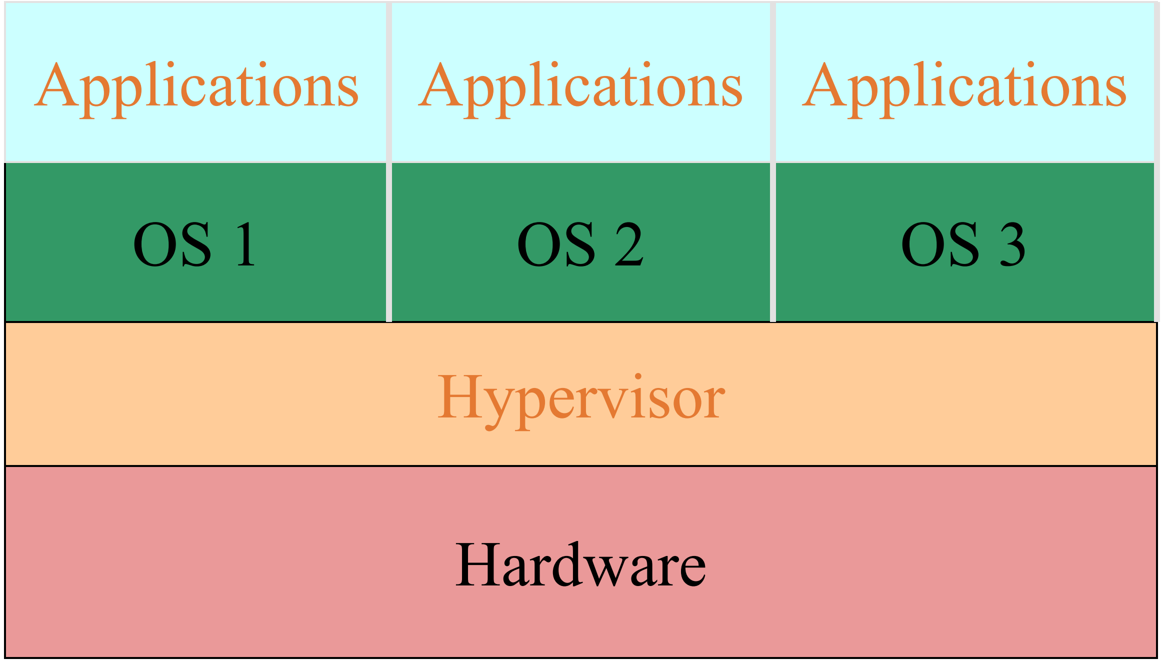
- 2007 moved to Vmware…. Started vCloud…

But there is way more

- Virtualize lots of machines – server consolidation
- Fault tolerance
- High availability
- Cloud computing
- Desktop consolidation – VDI
- …

It changed everything

Type 1 Hypervisor



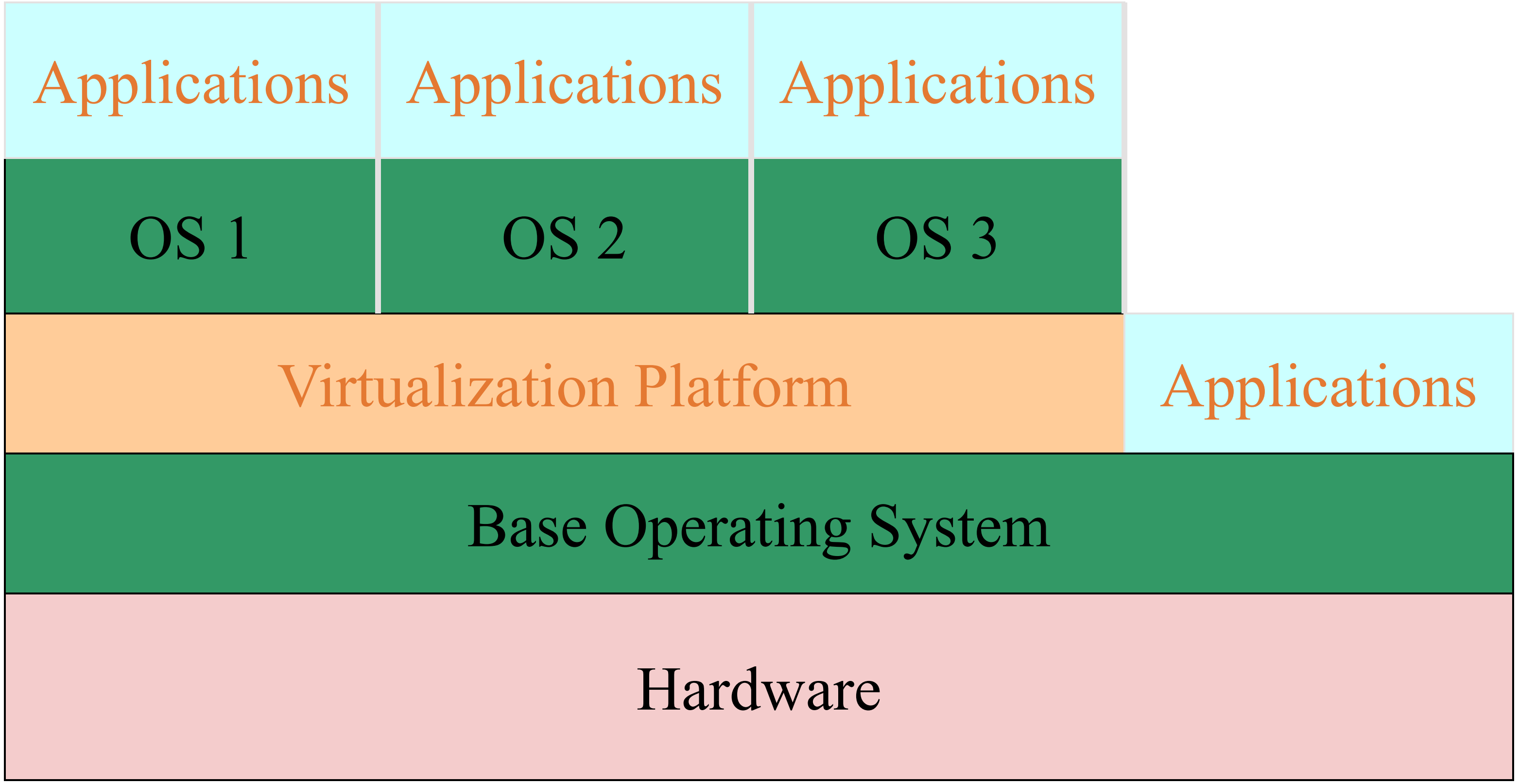
Applications Applications Applications

OS 1 OS 2 OS 3

Hypervisor

Hardware

Type 2 Hypervisor



Applications

Applications

Applications

OS 1

OS 2

OS 3

Virtualization Platform

Applications

Base Operating System

Hardware

What makes it hard?

- Isn’t VirtualBox just another program?
- No. User programs can only run non-privileged instructions

What makes it hard?

- Isn’t VirtualBox just another program?
- No. User programs can only run non-privileged instructions

```
$ cat foo.c
main()
{
    __asm__("movl    %eax, %cr3"::);
}
$ gcc foo.c
$ ./a.out
Segmentation fault (core dumped)
```

How to run an OS on top of another

- Attempt 1 - Emulate every instruction

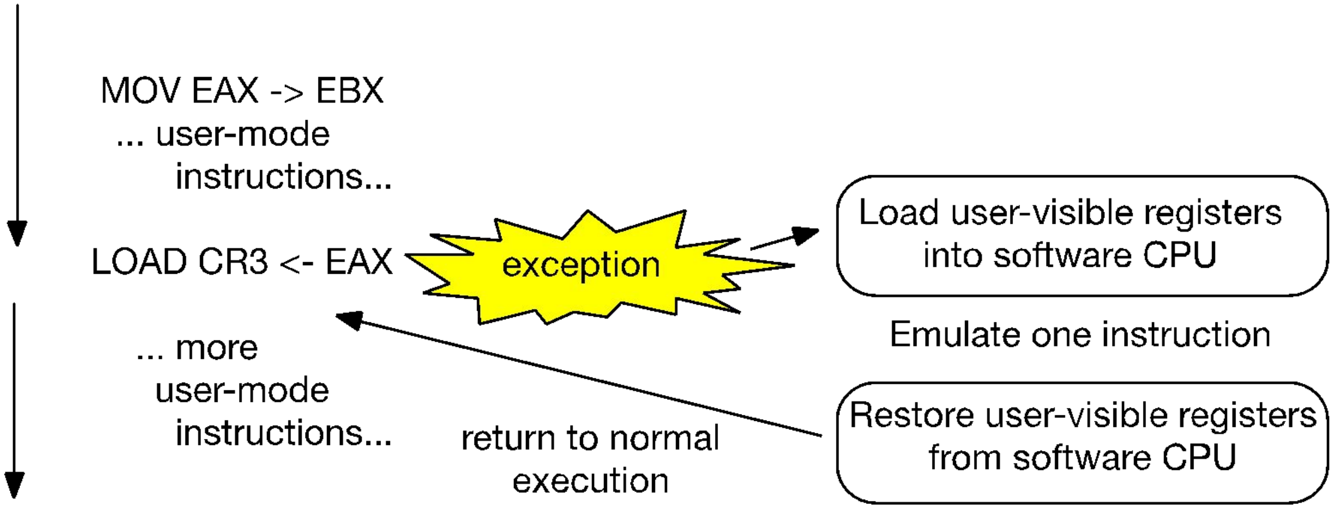
```
char memory[EMULATED_MEM_SIZE];
int R1, R2, R3, ...;
int PC, SP, CR1, CR2, CR3, ...;
bool S; /* supervisor mode */

while (true):
   instr = memory[PC];
   switch (instr):
     case "MOV R1 -> R2”:
       R2 = R1; break;
     case "JMP”:
       PC = … ; break;
     case "STORE Rx, <addr>”:
       <paddr> = TLB[<addr>]
       if <paddr> is real memory:
         memory[paddr] = Rx
       else
         simulate_IO_access(paddr, Rx)
     .... Etc. (for ~1000 more instructions)
```

Really slow (~100 cycles/instruction or more)

Trap and emulate (”classic” virt’n)

- Since you’re emulating the same CPU…
- Run everything in user mode
- When privileged instruction traps, load the software emulator, run for one step, load results back into CPU and continue direct execution


MOV EAX -> EBX
... user-mode instructions...
LOAD CR3 <- EAX
exception
Load user-visible registers into software CPU
Emulate one instruction
Restore user-visible registers from software CPU
return to normal execution
... more user-mode instructions...

But it doesn’t work ☹️

- Worked great on IBM machines from 1969 onwards
- But… x86 and ARM are not virtualizable CPUs
- Some of the privileged instructions don’t trap when you run them in user mode
  - Some are no-ops
  - Others do some but not all of what the privileged version does

Solution 1: Binary translation

- Guest user mode – direct execution
- When it tries to trap into the kernel, run all kernel code in emulation using BT

- That’s what got VMware where they are today.

Faster emulation through BT

- Binary translation = JIT compilation
- Translate code fragment X into code that does *what the emulator would do executing X*
- Typically expands the number of instructions executed (e.g. have to emulate MMU)
- Think of it as eliminating the loop and switch overhead in the emulator, plus you get to run an optimizer on the translated code.

- What Apple did for PPC->Intel switch (and 68k->PPC)
- ~3x-10x slower for good implementations

Example

```
ADD   R1+R2 -> R2
ADD   R2+R3 -> R3
MUL   2,R3  -> R3

          LOAD Rx <- &emulated_R1
          LOAD Ry <- &emulated_R2
          LOAD Rz <- &emulated_R3
          ADD Rx,Ry -> Ry
          ADD Ry,Rz -> Rz
          MUL 2,Rz  -> Rz
          STORE Ry -> &emulated_R2
          STORE Rz -> &emulated_R3
          RET
```

Virtualized Memory

- Problem – need 2 levels of translation
  guest virtual -> ”fake physical”
  “fake physical” -> real physical address
- But the CPU only has one level of translation
- Solution: fake page tables

Shadow page tables

- Emulated CPU (what the guest sees) has CR3 pointing to "fake" page tables
- Real CPU has CR3 pointing to real page tables
- On a real page fault, first check to see if there's a fake page table entry (guest virt -> fake phys)
  - If yes: calculate fake phys->real phys
    install guest virt -> real phys in real page table
  - Otherwise: pass the page fault to guest OS

I/O devices

- Device registers are just addresses in the physical address space
- So take page faults on them, and emulate what the real I/O device would do

- Can implement fake devices; e.g. disk in a file

Solution 2: Paravirtualization

- When you don’t really need full hardware virtualization

- Write an OS for running other operating systems
  - System calls are things like “add virtual memory mapping”

- Modify guest OS to run on it
  - Linux/arch/x86/xen/*

Paravirtualized I/O devices

- Everyone uses them
- No need to rewrite the OS – just load a device driver.
- Because taking page faults, disassembling instructions and grabbing values out of registers isn’t the most efficient API

Solution 3: new hardware

- 3 privilege modes:
  - User mode
  - Supervisor mode
  - Hypervisor mode

- All sorts of settings for when to trap into hypervisor mode.

While we're adding hardware…

- What about memory translation

- Add a 2nd set of page tables
- Guest page table:
  virtual -> "fake physical" address
- Hypervisor page table:
  fake physical -> real physical address

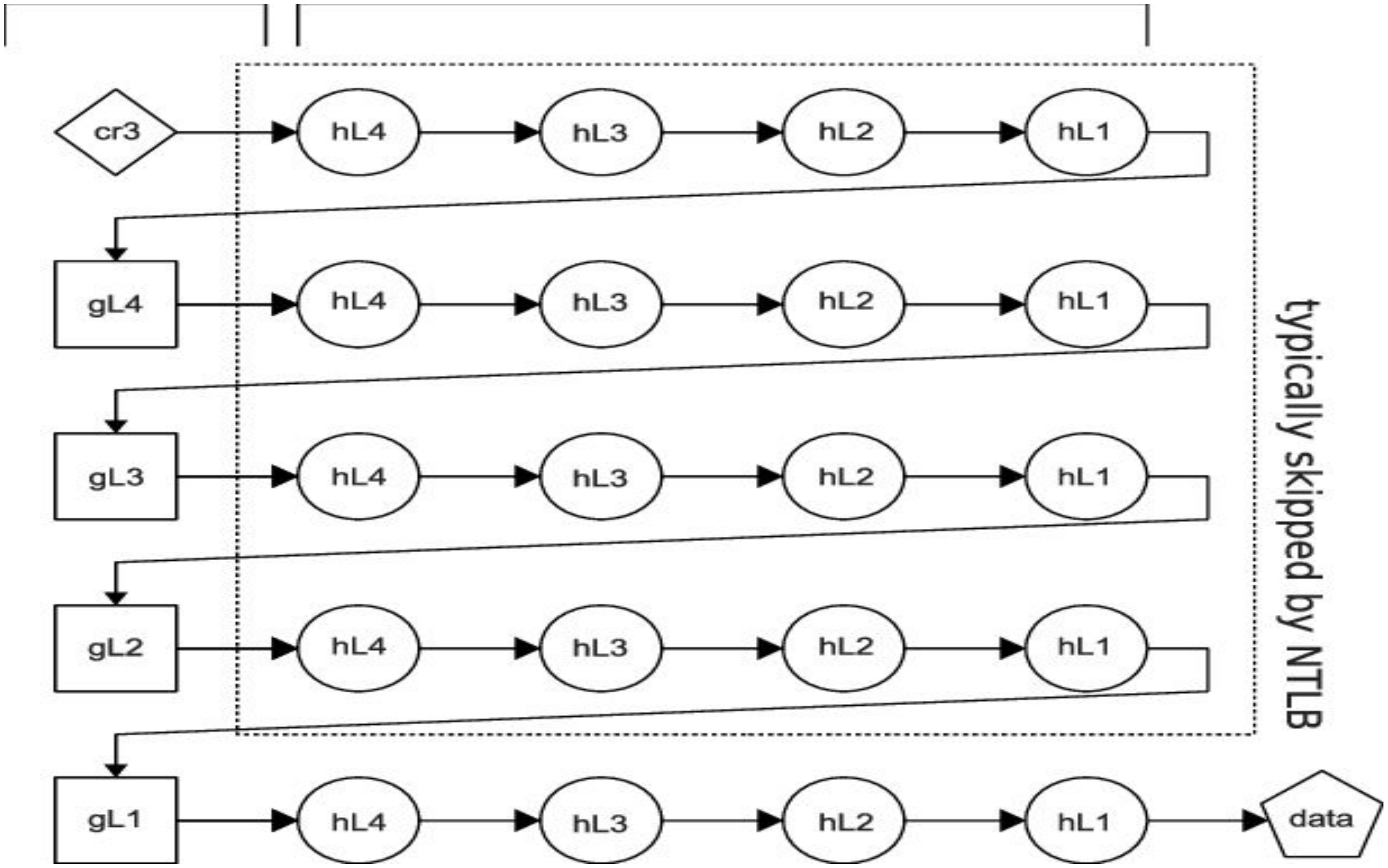
Ouch 16 levels of translation… thank god large pages

Alternative: nested paging

- With nested paging both guest and the hypervisor have their own copy of the processor state affecting paging such as the CR3.
- To avoid the software overheads under shadow paging, Intel x86_64 and AMD64 processors add Nested Paging and hardware page table walker hardware.
- Processors supporting nested paging maintain a Nested TLB which caches guest physical to system physical translations to accelerate nested page table walks
- Nested paging uses an additional or nested page table(NPT) to translate guest physical addresses to system physical addresses
- Nested page tables (nPT) map guest physical addresses to system physical addresses.
- When the page walk is completed, a TLB entry containing the translation from guest linear address to system physical address is cached in the TLB and used on subsequent accesses to that linear address.

walking nested page tables

http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf



But I/O still sucks…

- Paravirtualization still requires trap to hypervisor
- Lets throw more HW at the problem:
  - IOMMU – keep device from DMAing to wrong VM's memory
  - Single Root I/O Virtualization (SR-IOV)
    - Device exposes multiple set of registers mapped directly into VM – especially used for NIC that can support 100s
  - Share enough information that, if OS running, interrupt goes directly to it.

- This is just getting common today
- Last couple of years started to be used in the clouds with FPGA on NIC

Live Migration/vMotion

- Moving a VM from one computer to another
- Standard approach:
  - 1 copy pages while running, detect all modified pages
  - continue to copy until working set small
  - pause and finish copy
  - same as what we learned for fork
- Typically assumes network mounted storage
- Exploits standard networking techniques to advertise IP new location

Use cases

- Server consolidation - if load imbalanced, can move VMs around
- Dealing with server failures
- Move VMs and power off servers
- Key feature: Resource pools
  - shares/limits/reservation
  - enables administrator to control resource use across many VMs
  - key to increasing utilization of data center
- Not used in most of today's clouds

Concluding remarks

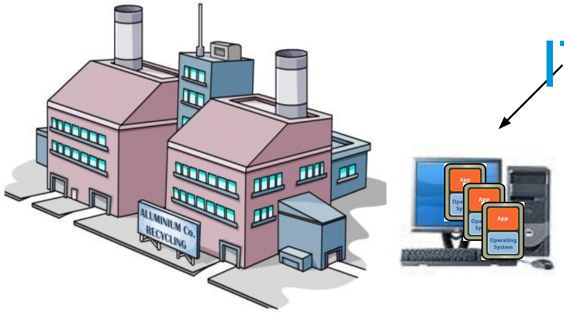
- Virtualization has transformed data center
  - Server consolidation, management, high availability, …
- Pervasiveness of virtualization resulted in hardware changing to support it.
- Today, with the most modern HW, virtualization is easy and there is little/no cost for virtualization.

Lessons I learned

1. A level of indirection solves a huge set of problems; VMware became a $69 Billion dollar company
2. Mendel/VMware had no idea what they where unleashing; released one product at a time
3. Sometimes success due to strange reasons:
   - Success of VMware due to GPL drivers
4. Huge problem for large companies to eat their own children
5. Get it working, then HW will catch up…

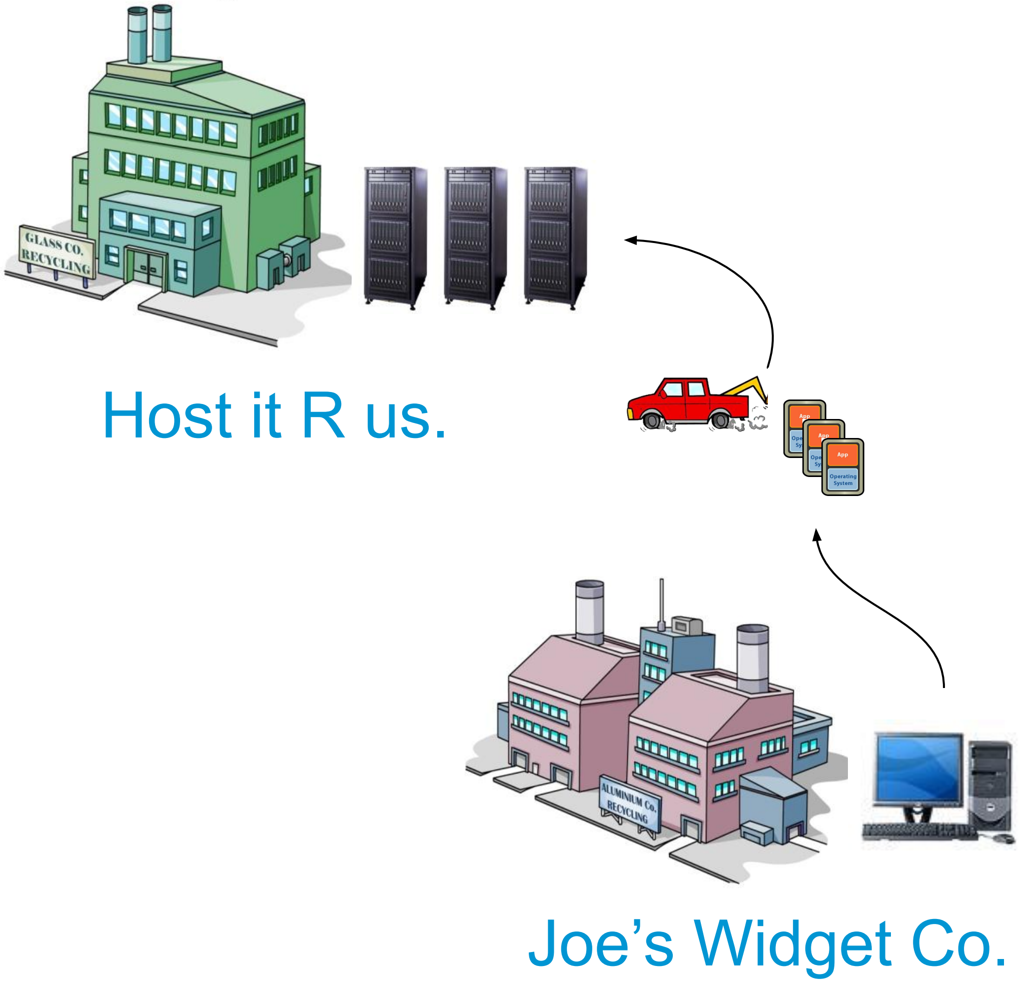
Value of Dell is $77B, and Dell owns 80% of VMWare

Cloud Computing



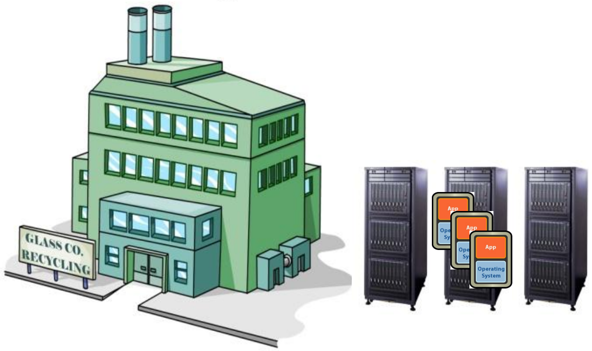
IT dept.

Joe’s Widget Co.

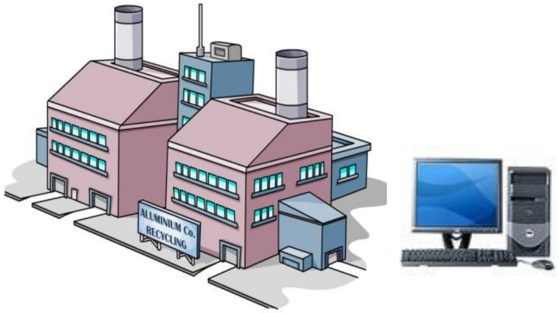


Host it R us.

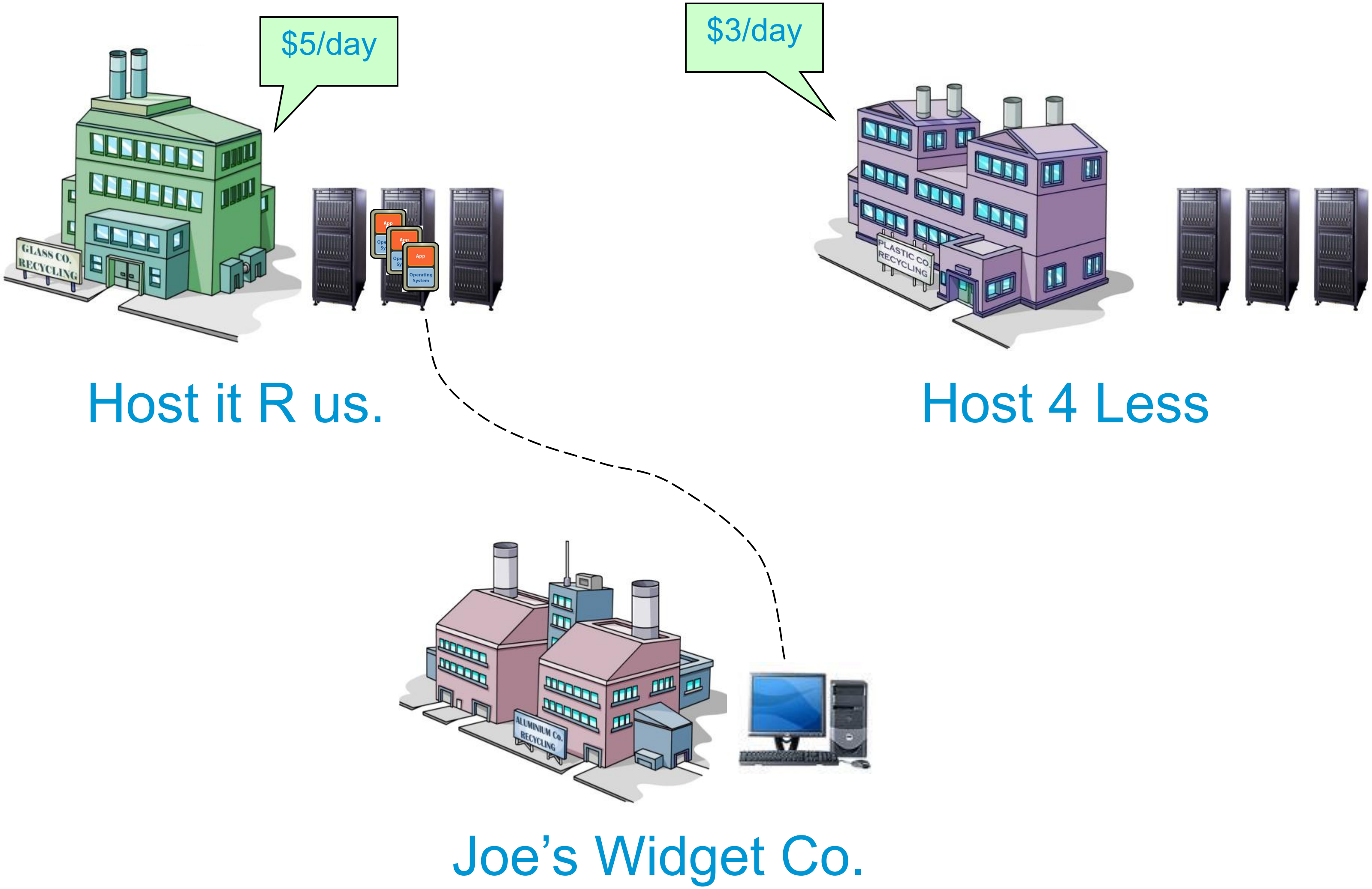
Joe’s Widget Co.



Host it R us.



Joe’s Widget Co.



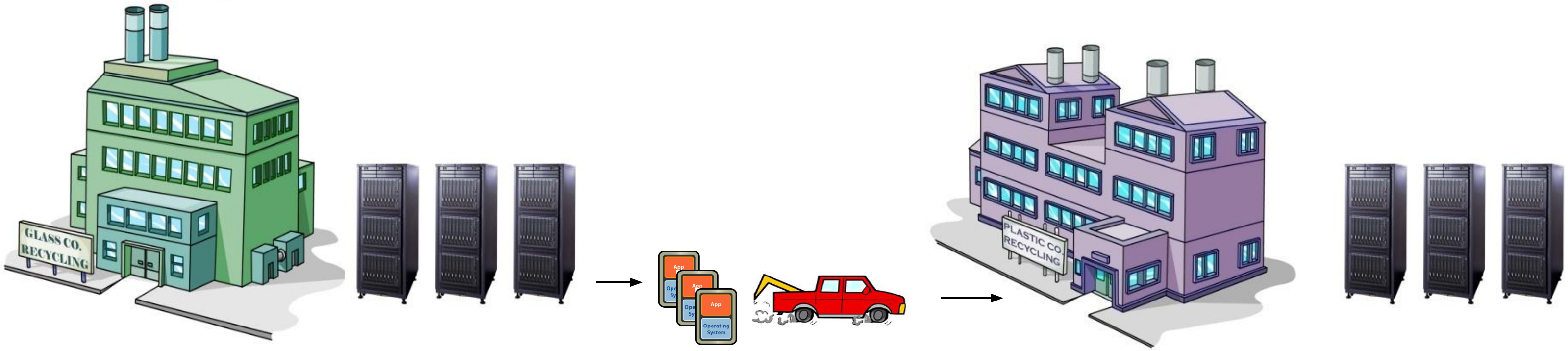
$5/day

$3/day

Host it R us.

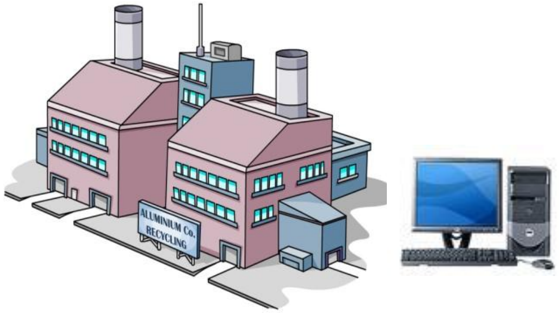
Host 4 Less

Joe’s Widget Co.

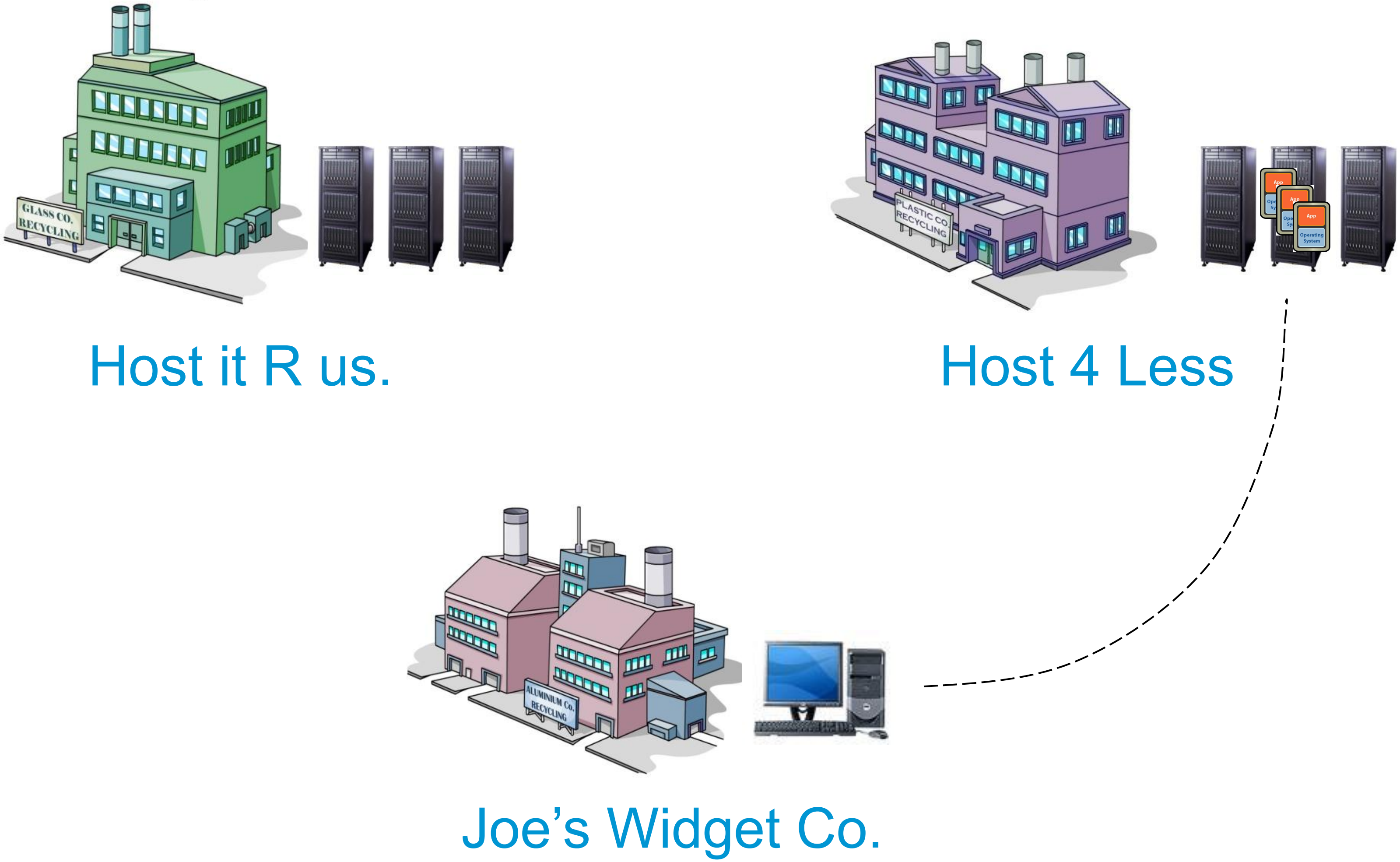


Host it R us.

Host 4 Less

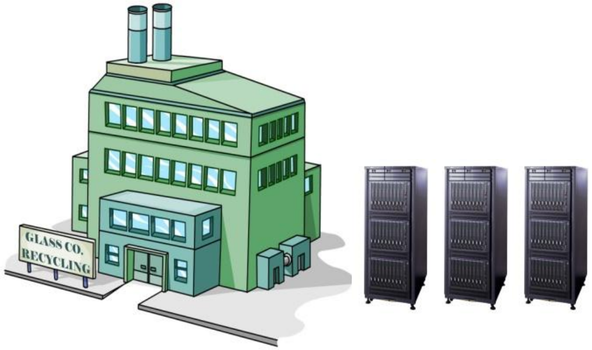


Joe’s Widget Co.

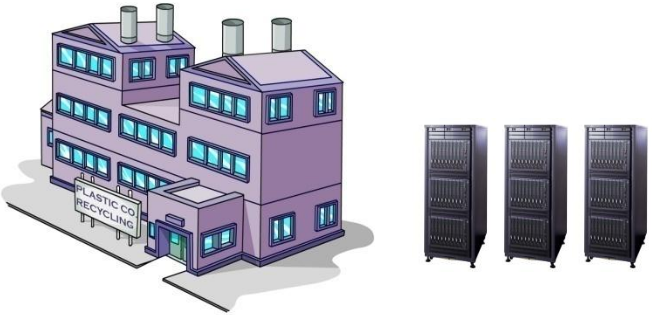

Host it R us.

Host 4 Less

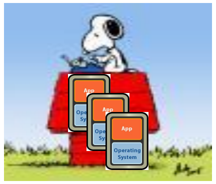
Joe’s Widget Co.



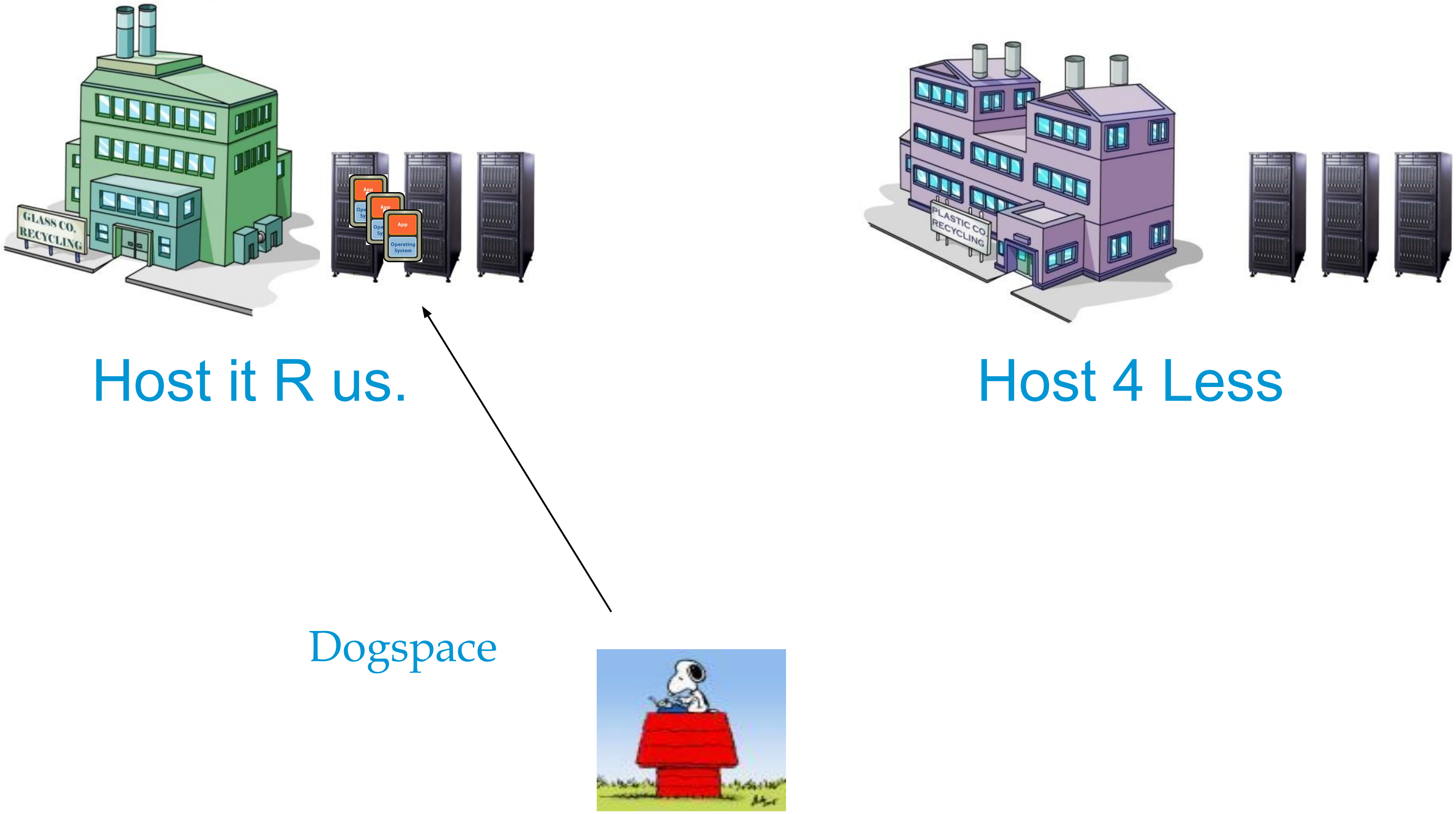
Host it R us.



Host 4 Less



Snoopy’s Startup

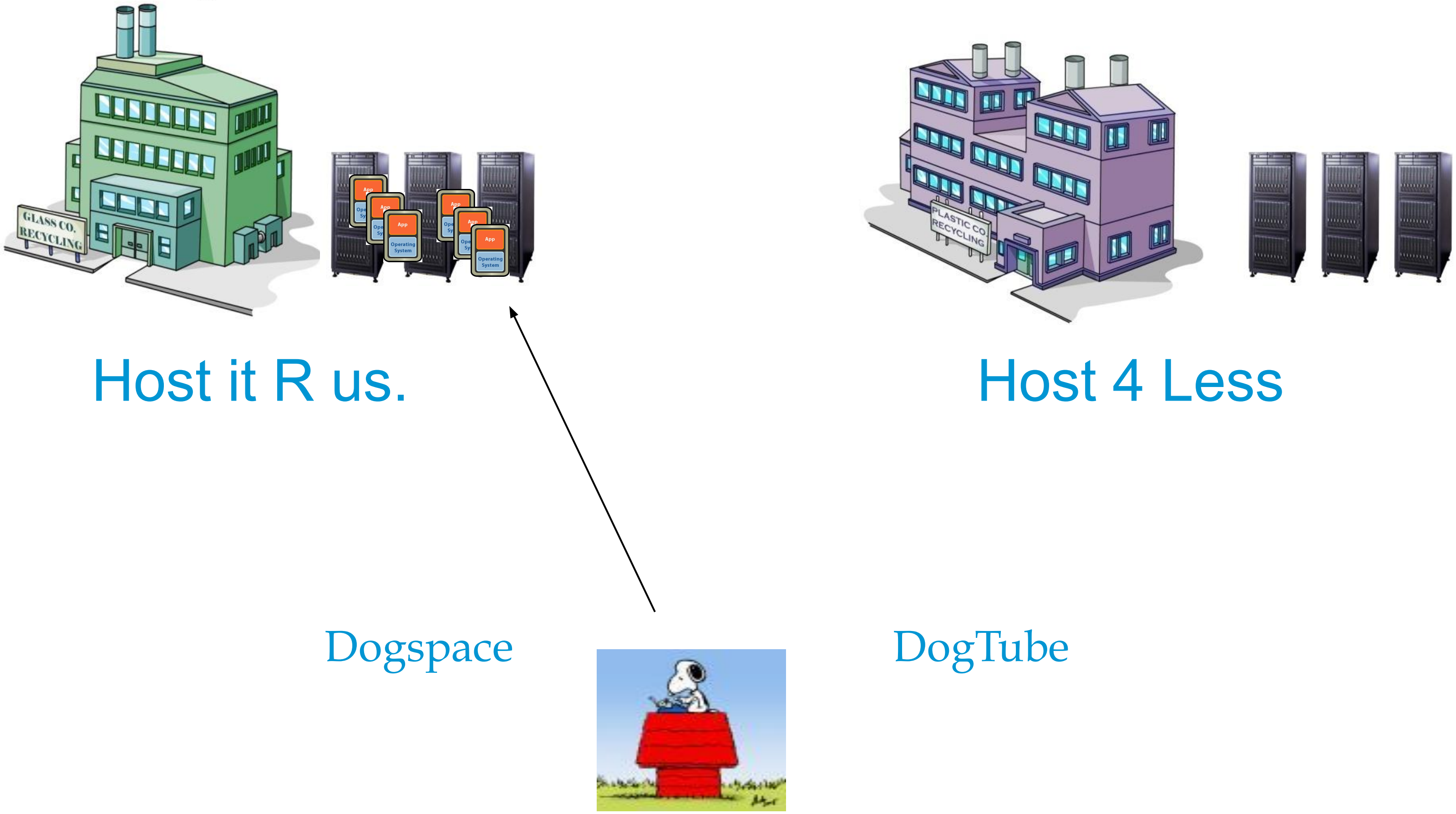


Host it R us.

Host 4 Less

Dogspace

Snoopy's Startup



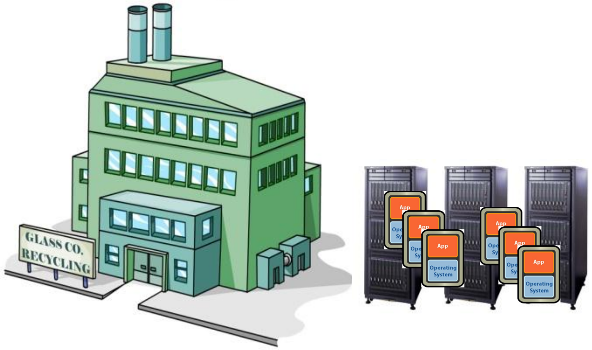
Host it R us.

Host 4 Less

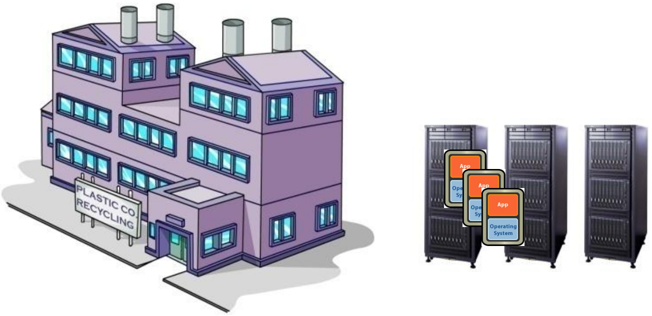
Dogspace

DogTube

Snoopy’s Startup



Host it R us.



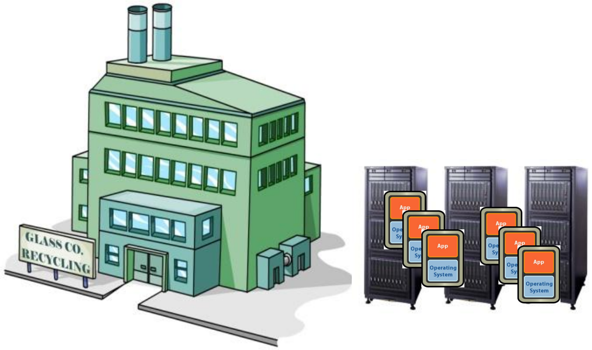
Host 4 Less

Dogspace

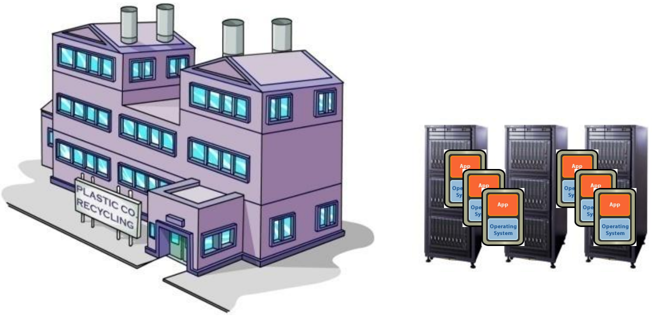
DogTube

DogFlix

Snoopy’s Startup



Host it R us.



Host 4 Less

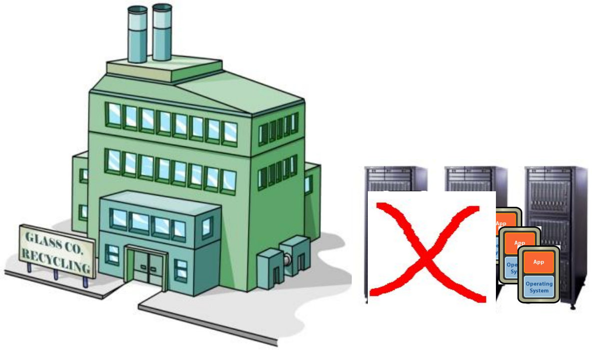
Dogspace

DogTube

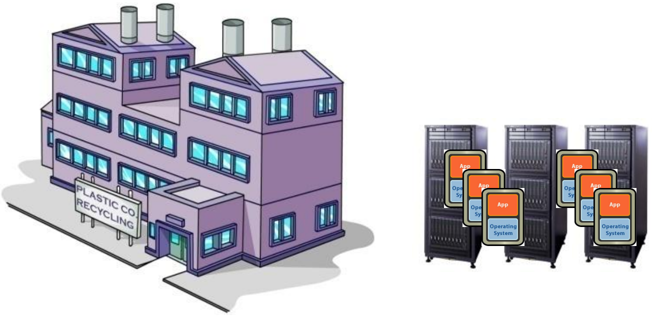
DogFlix

dBay

Snoopy’s Startup



Host it R us.



Host 4 Less

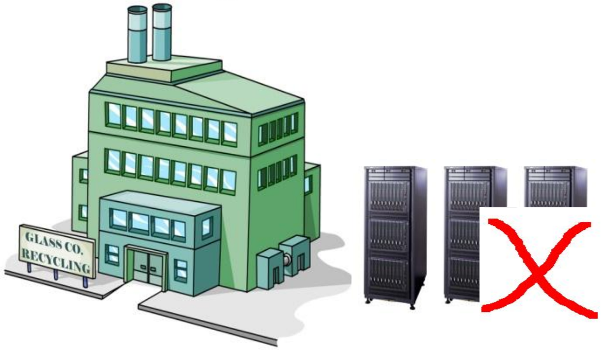
D ce

DogTube

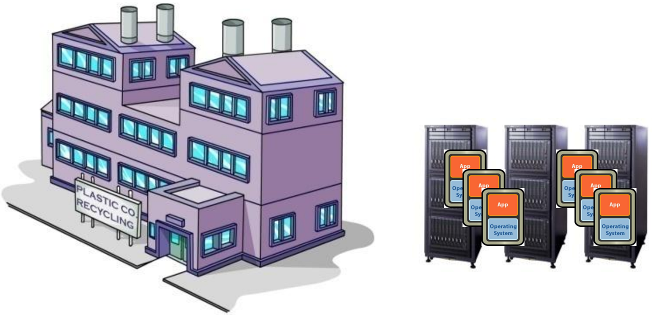
DogFlix

dBay

Snoopy’s Startup



Host it R us.



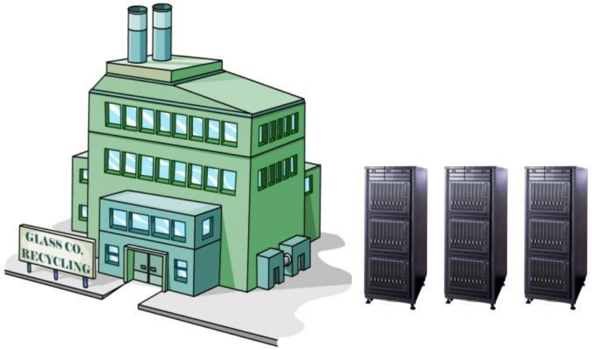
Host 4 Less

DogTube

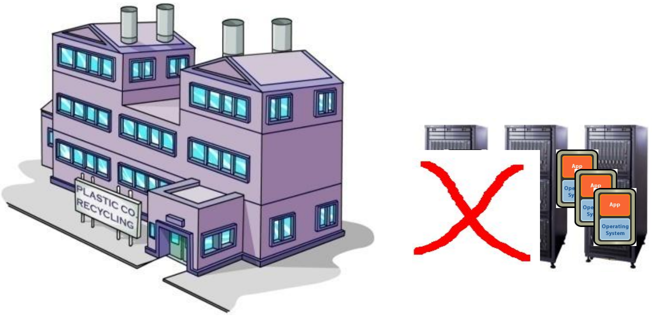
D k

dBay

Snoopy’s Startup



Host it R us.

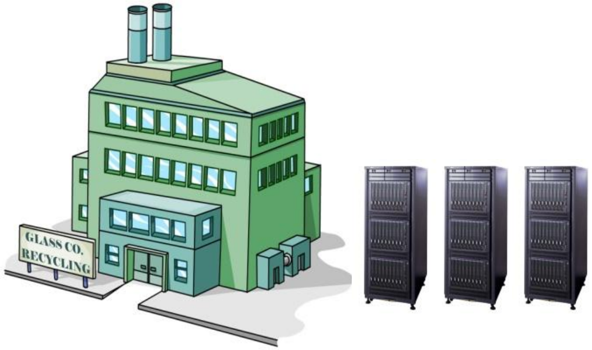


Host 4 Less

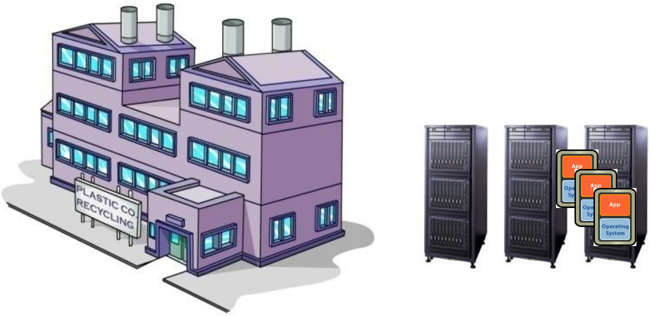
D e

dBay

Snoopy’s Startup



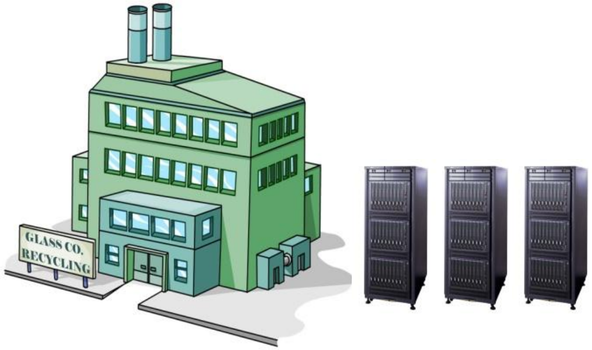
Host it R us.



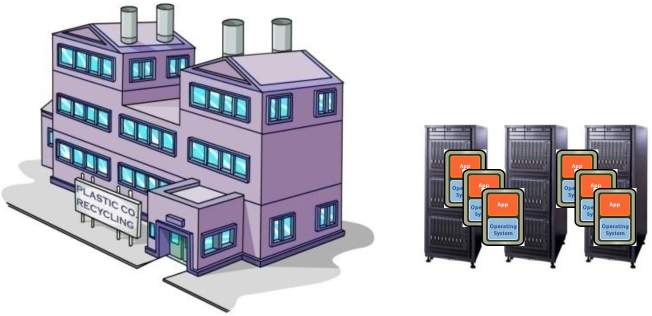
Host 4 Less

dBay ✓

Snoopy’s Startup



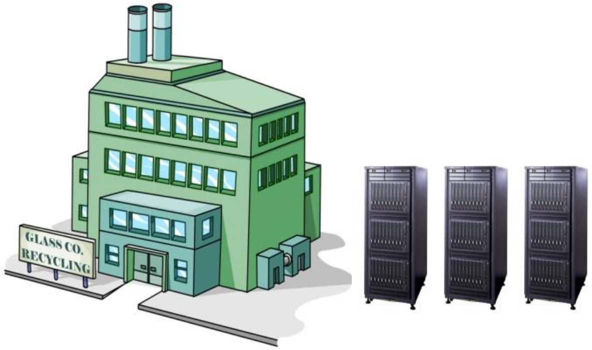
Host it R us.



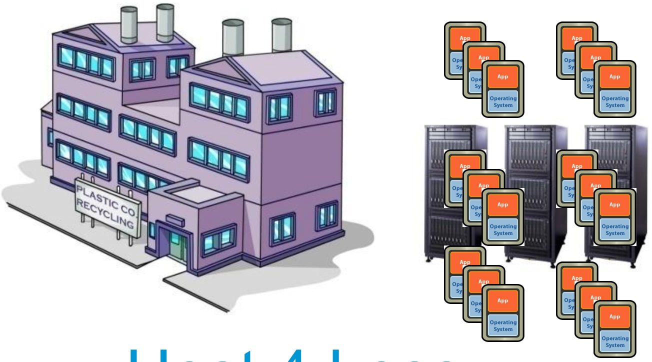
Host 4 Less

Snoopy's Startup

dBay ✓



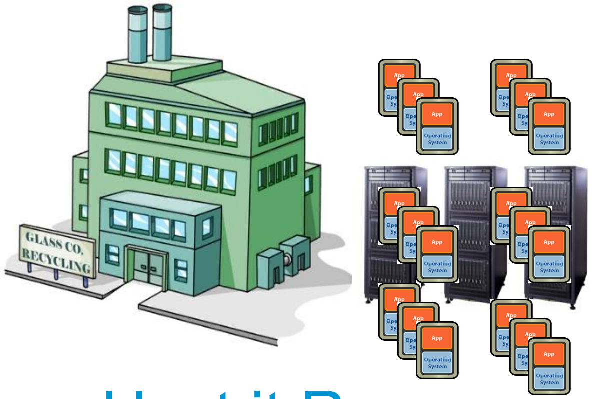
Host it R us.



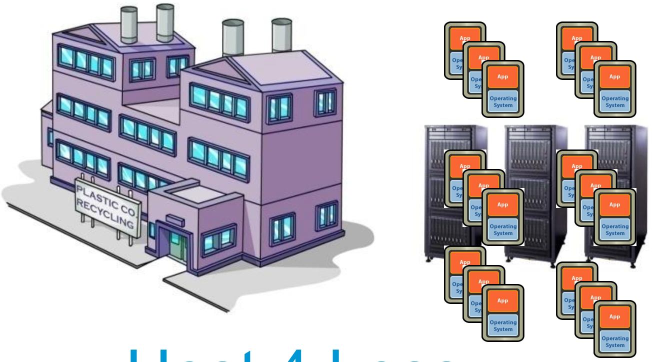
Host 4 Less

dBay ✓

Snoopy’s Startup



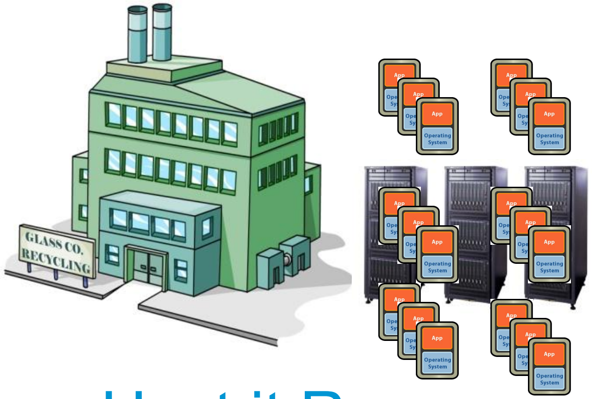
Host it R us.



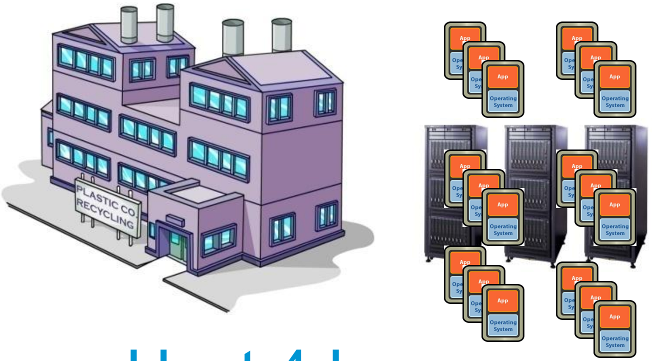
Host 4 Less

dBay ✓

Snoopy's Startup

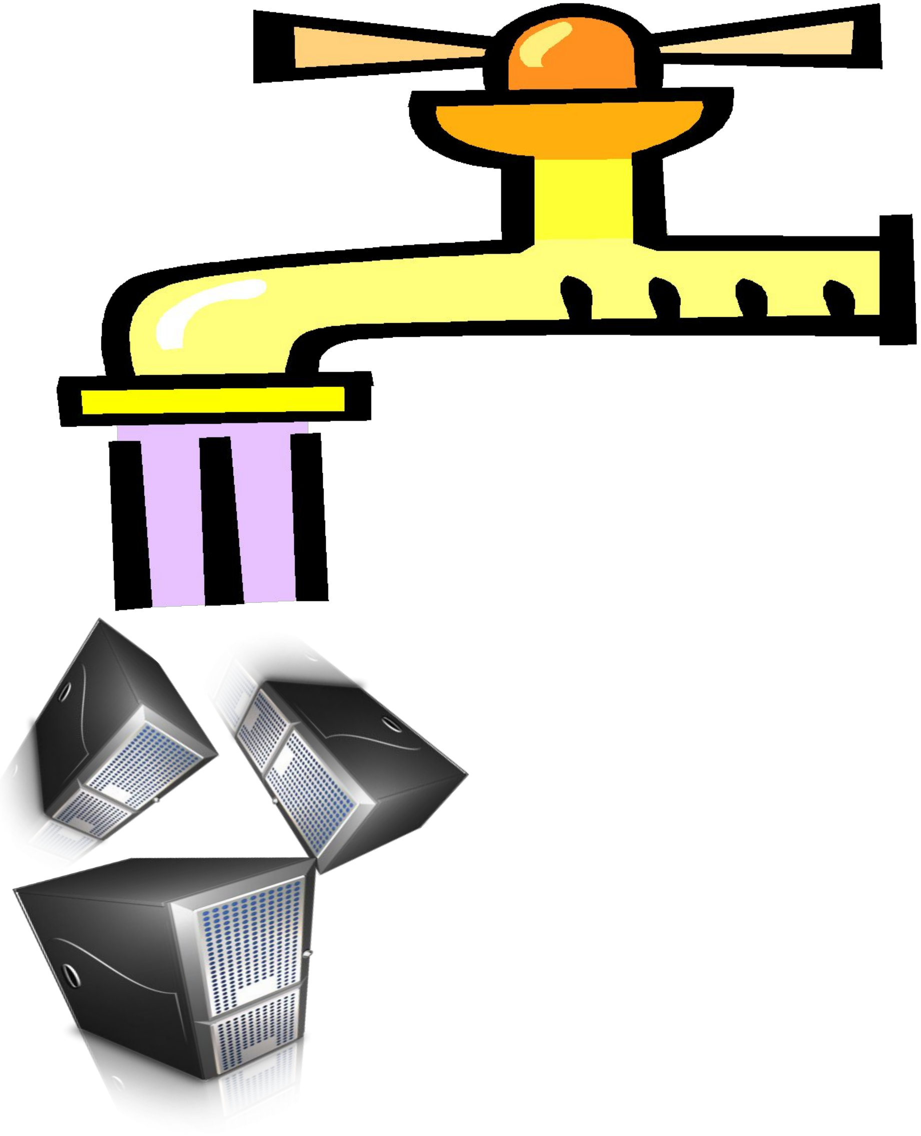


Host it R us.



Host 4 Less

What we want



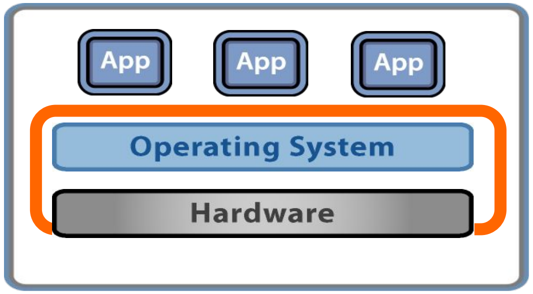
This is really nothing new…

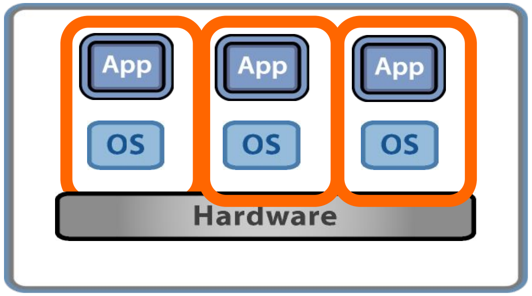
Original vision of Utility/grid computing:

"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."

When was this statement from?

Why now?





Why is this transformative

- Major change in computation is managed and used:
  - Economics of central utility: Price of computers, Operational efficiency, Location (e.g., cheap power, distribution), Co-location other customers, Utilization shared capacity, shared services (e.g., DR)
  - "As with the factory-owned generators that dominated electricity production a century ago, today's private IT plants will be supplanted by large-scale, centralized utilities." -- Nicholas Carr

- Availability of massive capacity on demand; elastically scale up and down:
  - Startups don't need to be acquired by Google or MS: a startup won't get money today to buy HW.
  - What happens when massive HPC becomes available to everyone?

- Gets rid of key impediments for developing & distributing SW
  - Avoids need for broad HCL, OS support, … many highly specialized software products⋯

Cloud in a nutshell

- On-demand access
- Economies of scale

All computing will
move to the cloud

Book just didn’t get it

- Virtualization key, but, most of today’s clouds don’t support migration
- More importantly, cloud has gone beyond just IaaS

Layers of cloud

- Infrastructure as a Service (IaaS): AWS, Azure, MOC, OpenStack, MOC…
- Platform as a Service (PaaS): Salesforce's Force.com, Google App engine, AWS, MSFT Azure
- Software as a Service (SaaS): hosted application: gmail, facebook, google docs, ebay

Motivation for using cloud

Is it about price?

- NO! cloud is 2-20x more expensive than local

The Real reasons:

- Administrators do not come in fractional units; if you are small cheaper
- Offers elasticity: can deal with massive fluctuations in demand
- Offers huge variety of services:
  - cloud provider can afford to amortize cost over a huge number of customers

Examples

- Microsoft's Azure
- Amazon's AWS
- Google's Cloud Services

Transformation

- Transformed how SW is developed:
  - continuous deployment; changes tested with real customers
  - Example facebook failure last year
  - massive advantage over waterfall
- Its all about distributed applications
  - change from pets to cattle
  - care about 99th% tail latency
  - stateless servers
  - huge set of higher level services: Containers as a Service, Functions as a Service, Analytics as a Service...

The challenges

- Monoculture from security perspective
- Emerging oligopoly:
  - Lack of competition limits sources innovation
  - Price is outrageously expensive
- Effort to lock in users: e.g., networking
- Big brother…, or perhaps just Giants whose incentives are not aligned with privacy and marketplace; Consider facebook

How does this impact OS?

- 99th percent tail latency changes what we care about
- Role of OS radically changing:
  - the cloud does isolation and resource management
  - containers and unikernels
- Enabling radically new HW: amortized over many users: FPGAs, GP-GPUs
- Exciting time to be doing OS research