# Cloud-temp

Josh Felmeden

October 5, 2021

# Contents

# 1 Virtualisation

The most fundamental type of cloud computing is IaaS compute, and the most fundamental type of this is the virtual machine. It has unmanaged services, which means you control what they do. You create, save or reuse them. They are networked and connected to storage and have certain security systems.

EC2 instances are a form of virtualisation.

- They mostly run on Xeon processors, but also have other processors available.
- There are lots of different tiers available on AWS that use different purposes
- They run in AMIs.
- After creating a virtual machine, you can start, stop, and terminate the machines. Once the machine is terminated, it will not come back.

In virtualisation, there is some virtual memory that points to addresses in physical memory. It is an abstraction of the storage resources, because the virtual memory seems contiguous, but in physical memory it could be in various locations. The operating system manages this and also has hardware support.

| **Virtual Memory** | **Virtual Machine** |
|---|---|
| Abstraction of the RAM memory resources | Abstraction of the storage/process/IO resources. |
| Mapping of program (virtual) memory addresses to physical addresses | Time slicing VM use of virtual memory addresses/CPU/IO registers in physical addresses. |
| Operating system manages | Operating system/Hypervisor manages |
| Hardware support (memory management unit) | Hardware support (e.g. Intel VT) |

## 1.1 The hypervisor

This virtualisation is not a new idea. It has been in use since around 1960, but it has been formalised in 1974, where they defined three important properties:

- *Fidelity*: Program gets the same output whether on VM or hardware
- *Performance*: Performs close to physical computer
- *Safety*: Cannot change or corrupt data on physical computer

This was ensured by analysing the instruction sets and identified two instructions that are the most important: **sensitive** instructions (that can change configurations of resources) and **privileged** instructions. Sensitive instructions need to be caught by the operating system, jumping from user mode to kernel mode. Therefore, all sensitive instructions need to be a subset of privileged instructions. This is all handled by the hypervisor.

Therefore, we have the hypervisor in contact with the server hardware, and then on top of this, we have the VM (or multiple machines) that consist of a guest OS, middleware, and apps. This type of VM is called **bare metal**.

Another type, called **hosted**, has a host OS running on top of the server hardware, and then the hypervisor.

The difference between the two is clear; in type two, the hypervisor can write to the host OS meaning that it can be more lightweight, where as type one can be much faster, at the cost of having to contain stuff that is normally handled by the OS.

### 1.1.1 Simulated Hardware

*Full virtualisation* is a complete or almost complete simulation of the underlying guest-machine hardware: virtualised guest OS runs as if it were on a bare machine.

The alternative to this is *paravirtualisation* and is only possible when the source code of the OS is available. The guest OS is edited and recompiled to make system calls into the hypervisor API to execute safe rewrites of sensitive instructions. In this example, the hypervisor doesn't simulate hardware.

EC2 offers both of these virtual machines:

- HVM — hardware virtual machine
  - Virtualised set of hardware
  - Can use OS without modification
  - Intel virtualisation technology
- PV — Paravirtualisation
  - Requires OS to be prepared
  - Doesn't support GPU instances

### 1.1.2 Xen

Xen is a free, open-source hypervisor developed at University of Cambridge. It has multiple modes:

- Paravirtualisation: guest OS recompiled with modifications
- Hardware assisted virtualisation: Intel x86 and ARM extensions
- Widespread
- Not easily virtualisable (17 instructions that violate the sensitive rules above)

### 1.1.3 Areas of Virtualisation

There are three main areas of responsibility for virtualisation managers:

- CPU virtualisation
  - Guest has exclusive use of a CPU for a period of time
  - CPU state of the first guest is saved, and the state of the next guest is loaded before control is passed to it
- Memory virtualisation

  - Additional layer of indirection to virtual memory
- I/O virtualisation
  - Hypervisor implements a device model to provide abstractions of the hardware

#### 1.1.4 KVM

Converts Linux into a type-1 hypervisor. Memory manager, process scheduler, I/O stack from Linux. The VM is implemented as a regular Linux process. KVM requires CPU virtualisation extensions to handle instructions.

#### 1.1.5 Nitro Hypervisor

- Based on KVM
- Offloads virtualisation functions to dedicated hardware and software
- Hypervisor mainly provides CPU and memory isolation to EC2 instances
- Nitro cards for VPC networking and EBS storage
  - Can handle NVMe SSD for instance and net storage, transparent encryption
  - Nitro hypervisor not involved in tasks for networking and storage
  - In OS Elastic Network Adapter driver
  - Security groups implemented in the NIC
- Can run bare metal

# 2 Auctions

## 2.1 Auction Theory

There are four common auction types:

- Open Ascending-Price (English)
- Open Descending-price (Dutch)
- First-price sealed bid
- Second-price sealed bid (Vickrey)

Auctions are not new, and have been used since the era of the Babylonians. Auctions are used because the *seller is unsure* about the values that bidders attach to the object being sold. If the seller knew bidder values, he could just offer the object to the bidder with the highest value $v$, or just below.

#### 2.1.1 Terminology

**Private values**:

- Each bidder knows the value of the object to themself at the time of the bidding

- No bidder knows with certainty the values attached by *other* bidders, and knowledge of the other bidders' values have no effect

This assumption is most plausible when the value of the object to a bidder is derived from its consumption or use alone, such as a pizza or a bicycle.

**Interdependence** is when the object worth may be *unknown* at the time of the auction to the bidder.

- Bidder may have only an *estimate* or some private signal that is correlated with the true value.
- Other bidders may possess *information* that, if known, would affect the value that a particular bidder attaches to the object.

We call this specification **interdependent values** which are values that are unknown at the time of the auction and may be affected by the information available to other bidders.

This assumption is most plausible for situations in which the object being sold that can possible be resold after the auction, such as a Harry Potter first edition book.

### 2.1.2 Equivalent Auctions

- **First-price sealed-bid auction**:
  - The bidder's strategy maps private information to a bid
  - No information about other bidders is available as the auction is sealed
  - The bidder needs to consider what the bid, as they have no idea what other people might bid
- **Dutch auction**:
  - Conducted in the open, but offers no useful information to bidders
  - The only information available is that some bidder has agreed to buy at the current price, but that causes the auction to end.

These two are equivalent because bidding a certain amount in the FPSB auction is equivalent to offering to buy at that amount in a Dutch auction. Therefore, Dutch open descending price auction is strategically equivalent to the first-price sealed-bid auction.

**strategic Equivalence**: for every strategy in one game, a player has a strategy in another game that results in the same outcomes

- **English auction**
  - Offers information when other bidders drop out. By observing this, it may be possible to infer something about their privately known information
  - with *private values*, however, this information is of no use
- **English Auction Strategy**
  - It cannot be optimal to:
    - Stay in after the price, $p$ exceeds the value $v$ (incurring a loss)
    - Drop out before the price reaches the value (losing out on potential gains)
  - Therefore, the optimal strategy is to bid up to the value $p = v$

the strategy for a **Second-price sealed-bid auction** is also the same:

- Bidder $B$ with private value $v$ bids a price $p$.
- The highest competing bid has price $c$

So, the optimal bidding strategy for this auction is:

- If $B$ bids at $p = v$:
  - $B$ wins if $v = c$ (making profit $v - c$) and does not win if $v < c$.
  - If $v = c$, assume $B$ is indifferent between winning and losing
- Alternatively, if $B$ bids $p$ lower than $v$ (i.e. $p < v$):
  - If $v > p > c$, then $B$ still wins and profit is still $v - c$
  - If $c > v > p$, then $B$ still loses, so also no change, BUT
  - If $v > c > p$, the $B$ now loses, so makes less profit
  - Therefore, bidding $p < v$ can never increase profit for $B$ but can decrease profit.

We say that these auctions are **weakly equivalent** because the two auctions are not strategically equivalent, but the optimal strategies are the same if the values are private.

With interdependent values, the information available to others in the open auction is relevant to a bidder's evaluation of worth. Seeing some other bidder drop out early may make the bidder realise that his own estimate is too great. Therefore, if the values are interdependent, the two auctions may not be equivalent from the perspective of the bidders.

An auction is said to be *incentive compatible* if it encourages bidders to bid their *true value of the good*

Both English and Vickrey are incentive compatible, since when values are private, the optimal strategy is to bid $p = v$.

However, Dutch and FPSB are not, since we want to bid lower than the true value; attempting to stop the auction just below the price of what other bidders value the good at.

Incentive compatible auctions stop game-playing between bidders.

### 2.1.3 Revenue Equivalence Theorem (RET)

- In English/Vickrey auctions, bidders bid price $p = v$.
- In Dutch/FPSB, bidders price $p < v$.
- However, the expected revenue in a first-price auction is the same as the expected revenue in a second-price auction.

The revenue equivalence theorem states that if the values are private, and bidders are risk neutral, any standard auction will yield the **same expected revenue** to the seller.

The revenue equivalence theorem requires *risk neutral* bidders, meaning that they have no emotional attachment to the bids and they only seek to maximise expected profits. However, in reality, bidders are normally *risk averse*, meaning that they will bid higher, as they will buy *insurance* against the possibility of losing. When we take risk aversion into account, we get the result that the expected equivalence in the first-price auction is greater than that in a second-price auction.

The RET also requires private values, but in reality, the values are often interdependent, since we often change our valuations based on valuations from others. When we take interdependence into account, ordinary ascending auctions are more profitable than standard (first-price) auctions.