

# **CS 3502**

# **Operating Systems**

## **Miscellaneous**

**Kun Suo**

Computer Science, Kennesaw State University

<https://kevinsuo.github.io/>

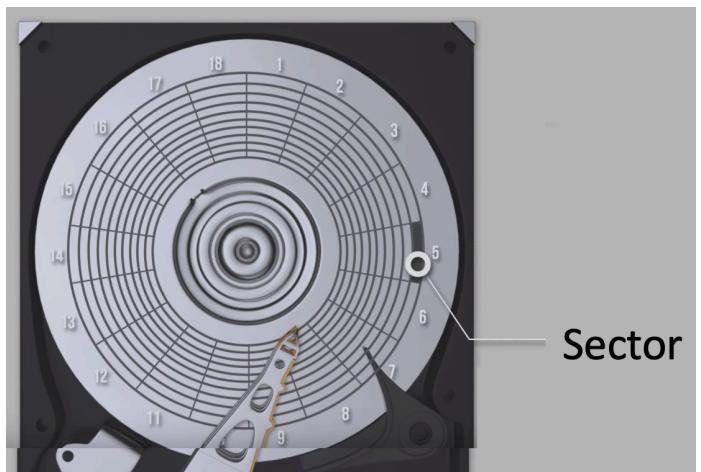
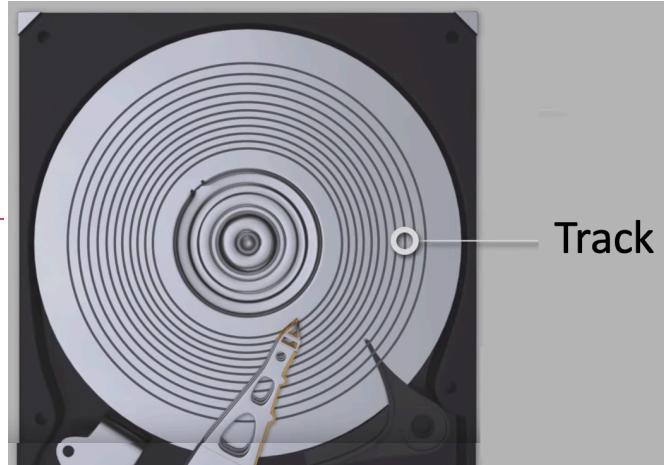
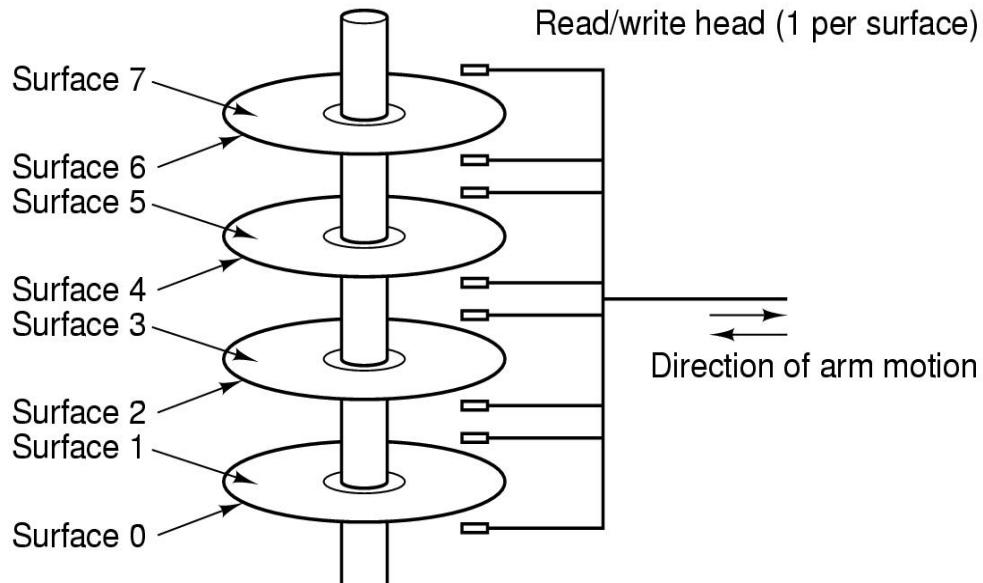
# Outline

---

- Disk scheduling algorithm
- Time in OS and Networking
- Power management in OS
- Virtualization and cloud

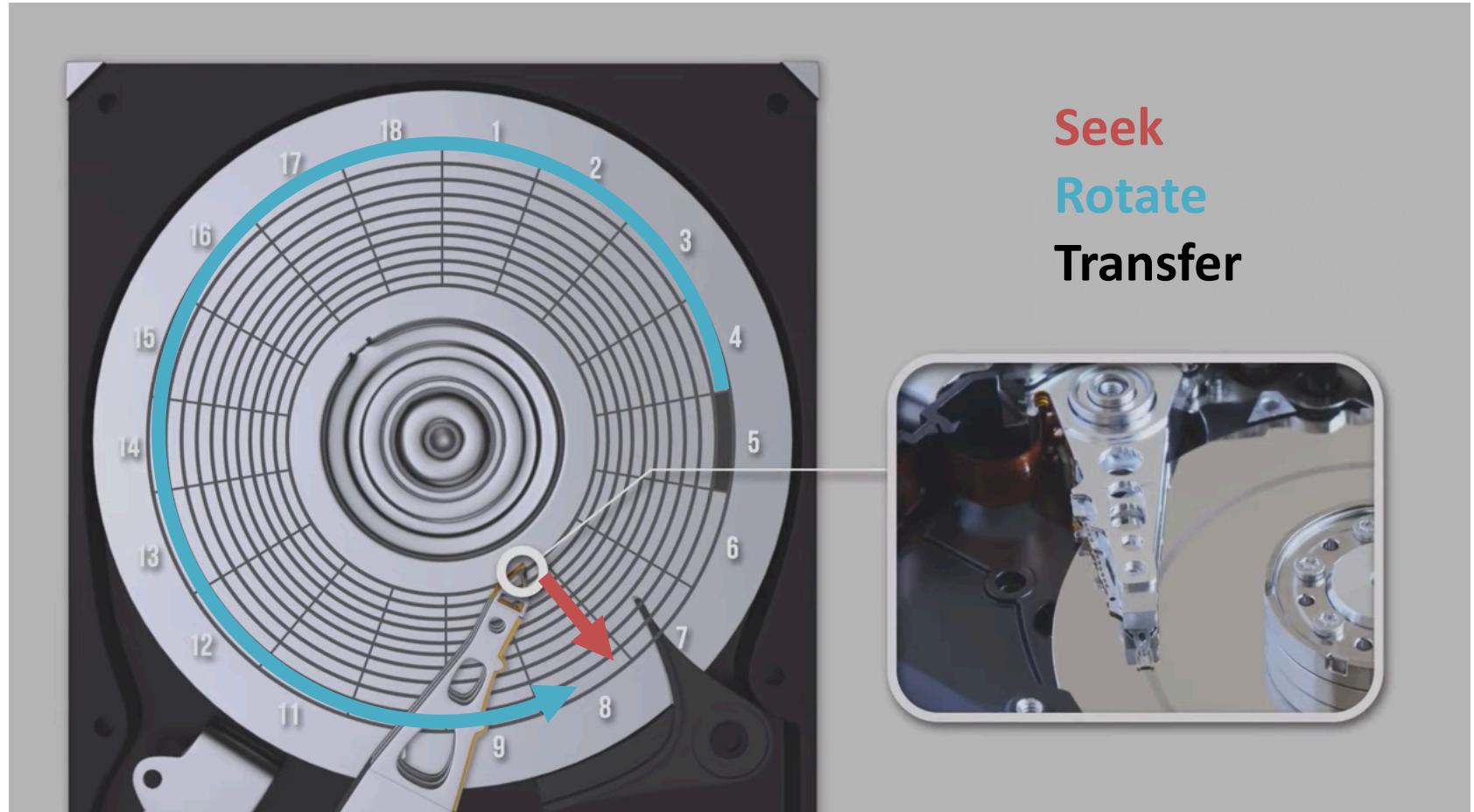


# Hard Disks



- The conventional *sector-track-cylinder* model
  - A stack of platters, a surface with a magnetic coating
  - Magnetic disks are organized into cylinders, for R/W

# Three-Stage Disk R/W Process



# Three-Stage Disk R/W Process

---

- Three-stage Disk R/W process: time required to read or write a disk block determined mainly by 3 factors
  1. Seek time (Dominates), 10-100ms
  2. Rotational delay, 1-10ms
  3. Actual transfer time

**Disk Access Time = Seek time + Rotational Latency + Transfer time + others**

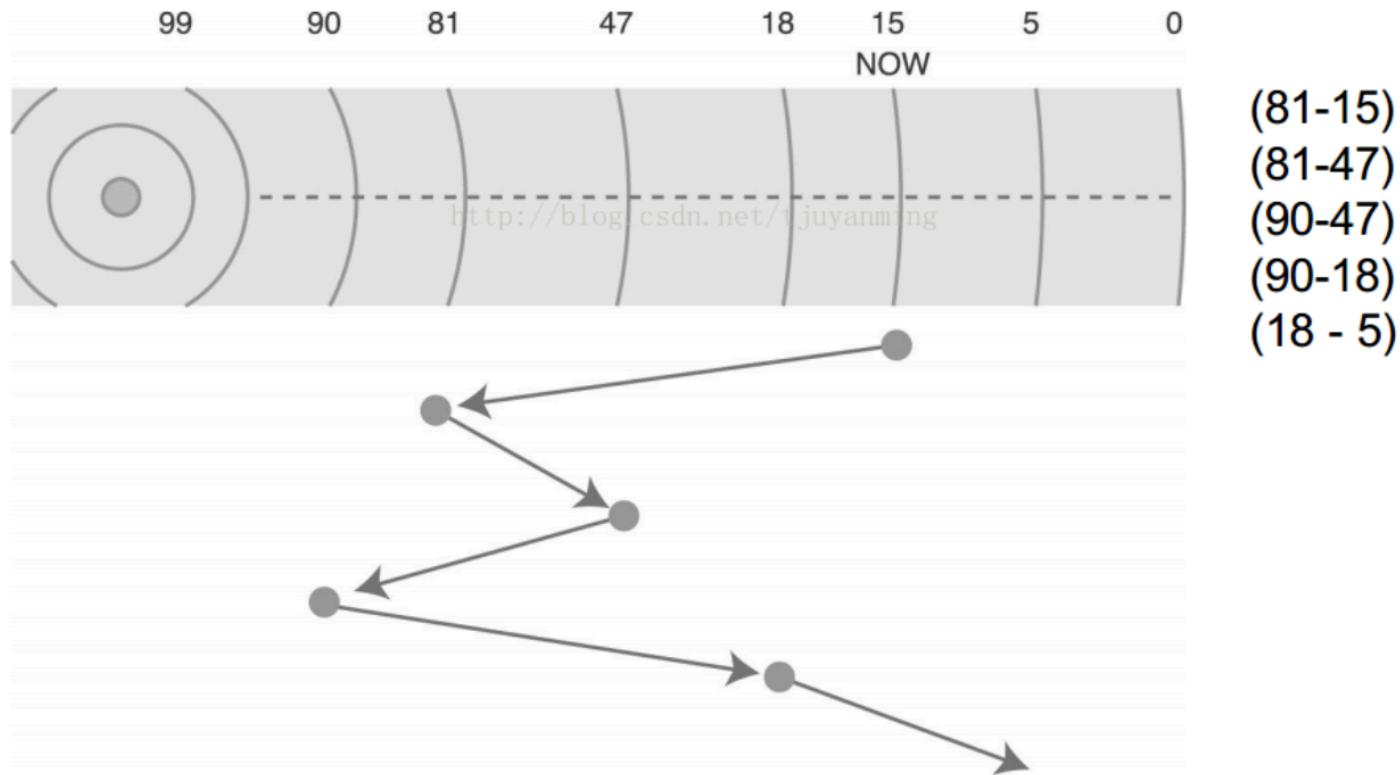
**How to optimize seek time?**

**Disk arm scheduling algorithms**



# Disk scheduling algorithm: FCFS

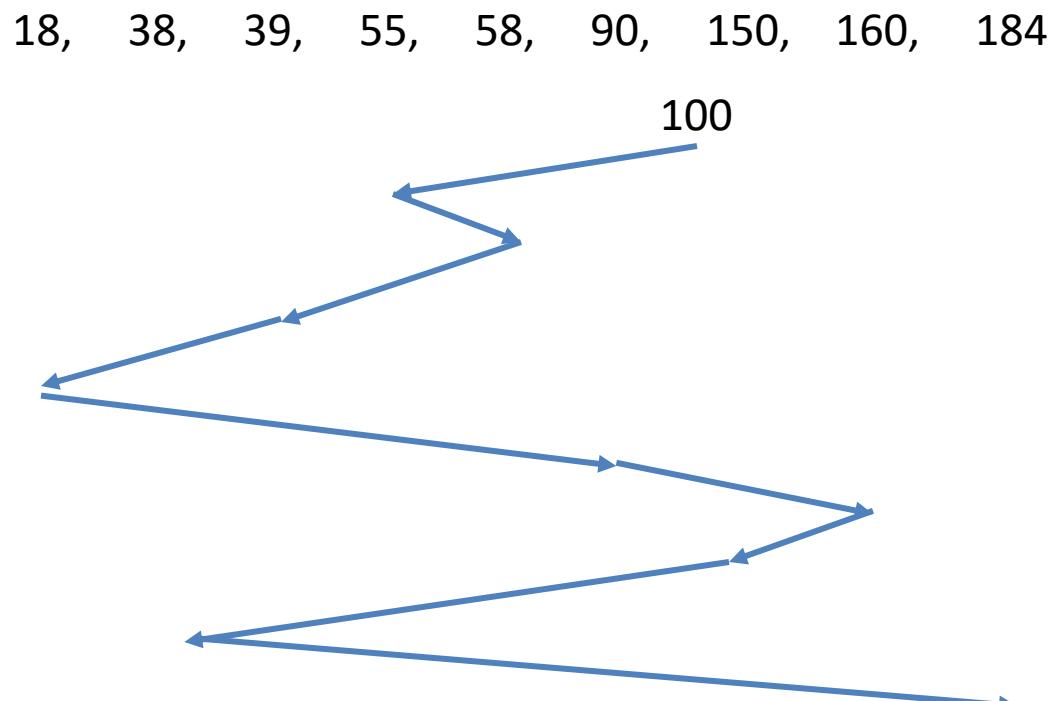
- FCFS: scheduling in order
- 81, 47, 90, 18, 5;      NOW at 15



# Disk scheduling algorithm: FCFS

- Example: header start from 100 and the track sequence is 55, 58, 39, 18, 90, 160, 150, 38, 184

Start from 100	
The next track	Distance
55	45
58	3
39	19
18	21
90	72
160	70
150	10
38	112
184	146
Average: 55.3	



# Disk scheduling algorithm: FCFS

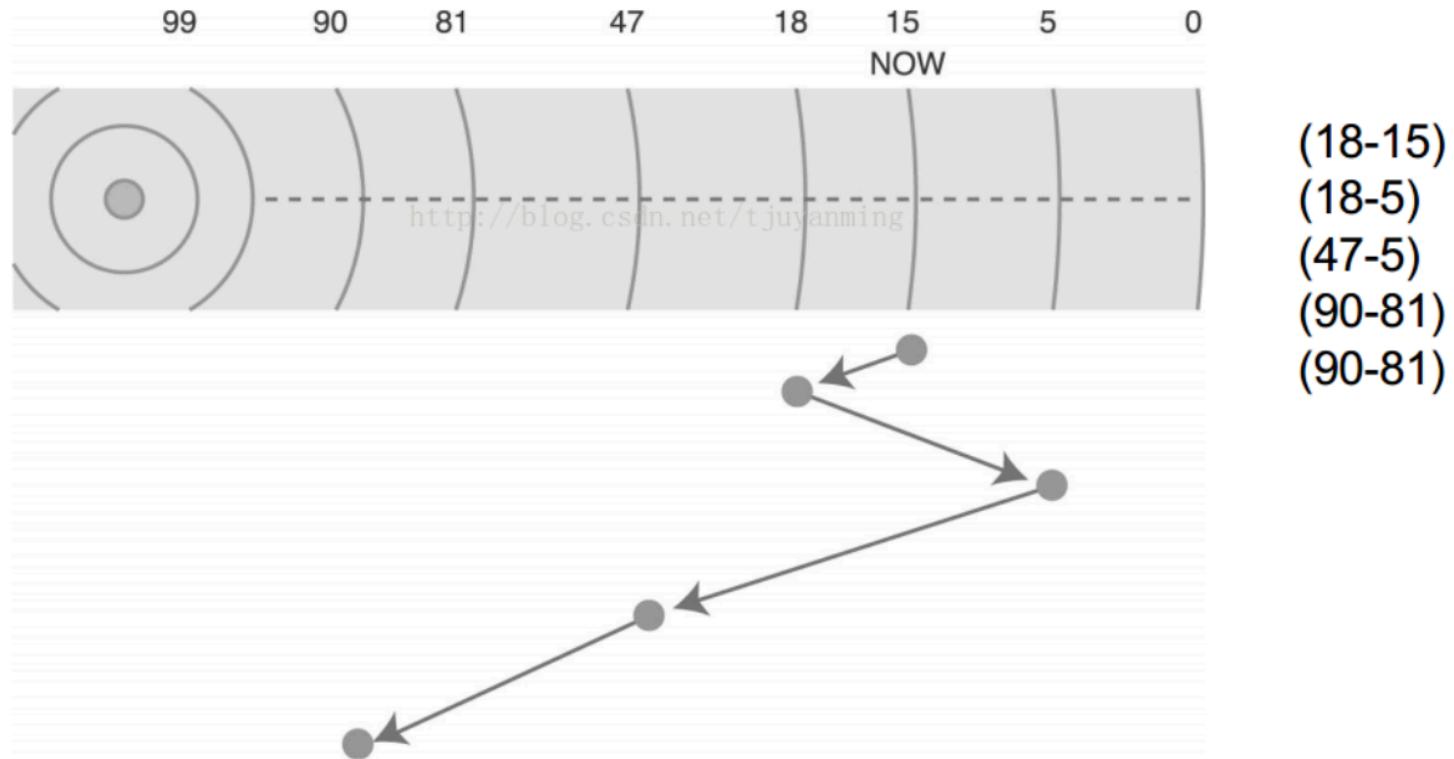
---

- Pros:
  - Fair and simple
  - each process request can be processed in turn
- Cons:
  - The average seek time is long
  - suitable for occasions where the number of disk I/O processes is small



# Disk scheduling algorithm: Shortest Seek First (SSF)

- SSF: scheduling to the track which is the closest
- 81, 47, 90, 18, 5;                    NOW at 15

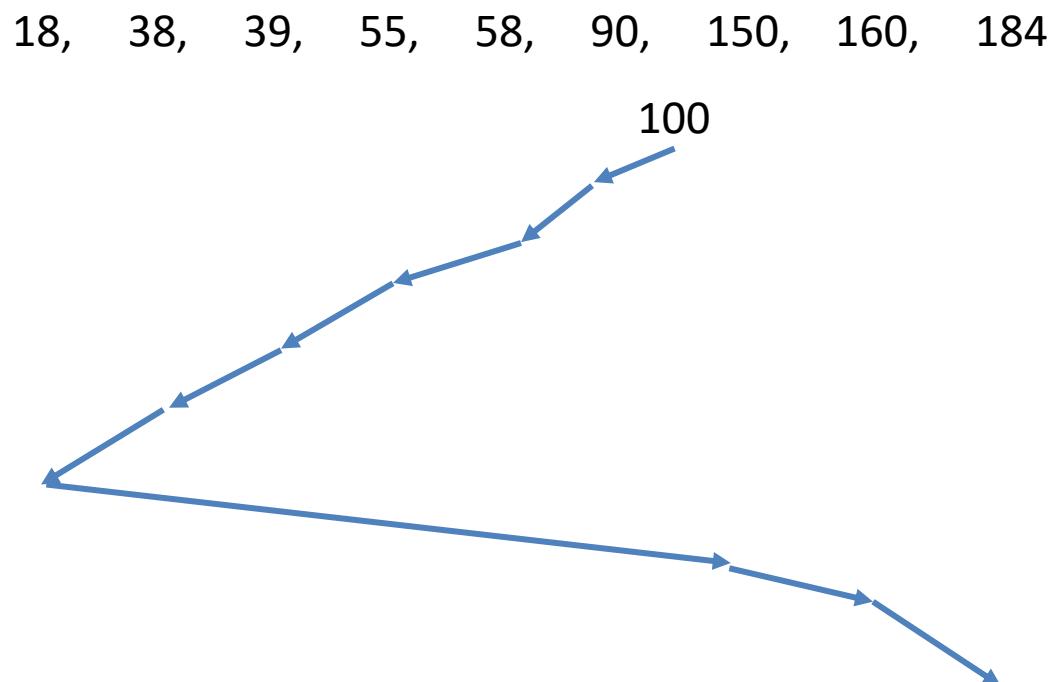


# Disk scheduling algorithm: Shortest Seek First (SSF)

- Example: header start from 100 and the track sequence is 55, 58, 39, 18, 90, 160, 150, 38, 184

Start from 100	
The next track	Distance
90	10
58	32
55	3
39	16
38	1
18	20
150	132
160	10
184	24

Average: 27.5



# Disk scheduling algorithm: Shortest Seek First (SSF)

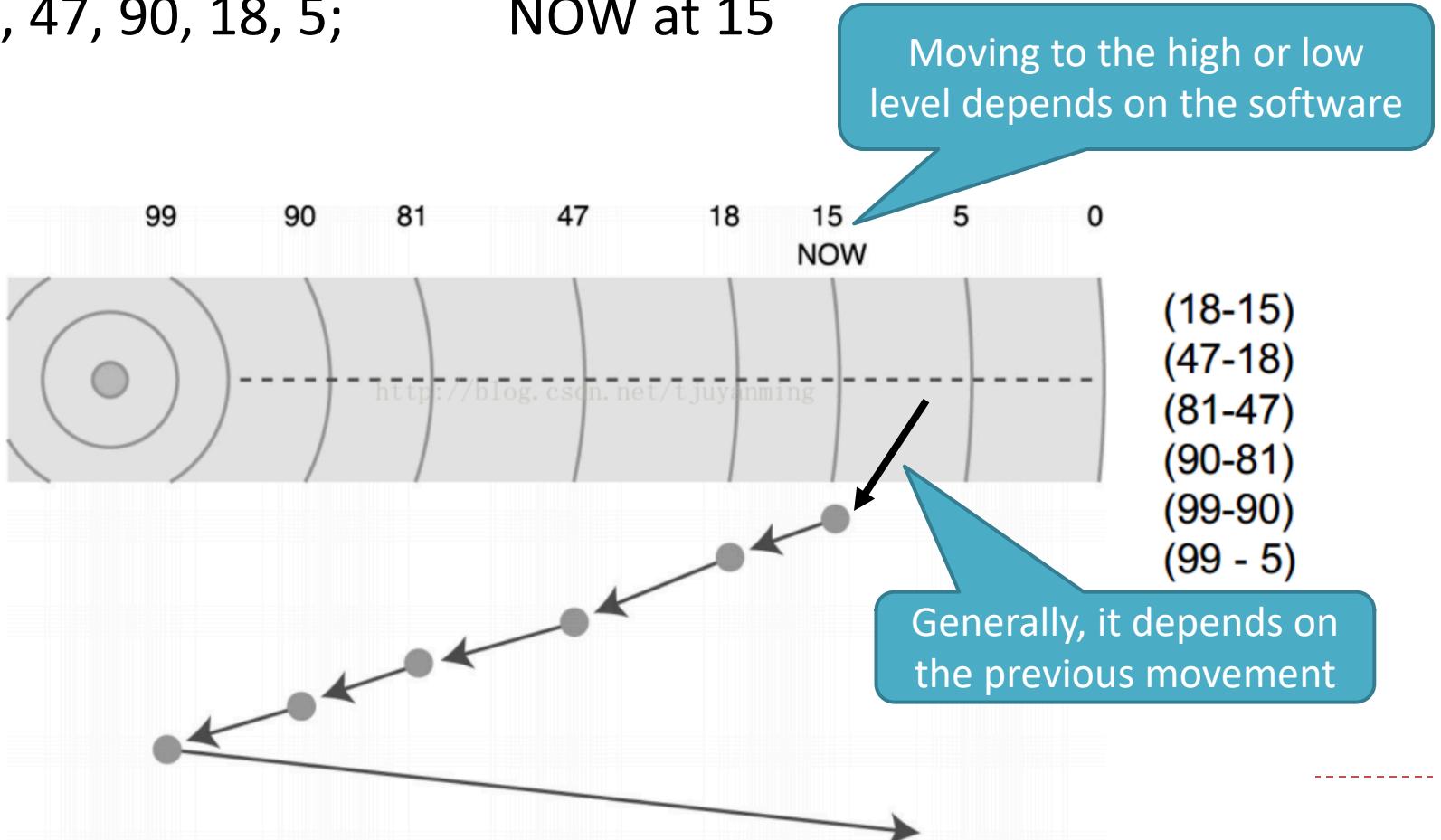
---

- Pros:
  - Schedule the track closest to the header
- Cons:
  - Starvation occurs in low priority processes



# Disk scheduling algorithm: elevator algorithm

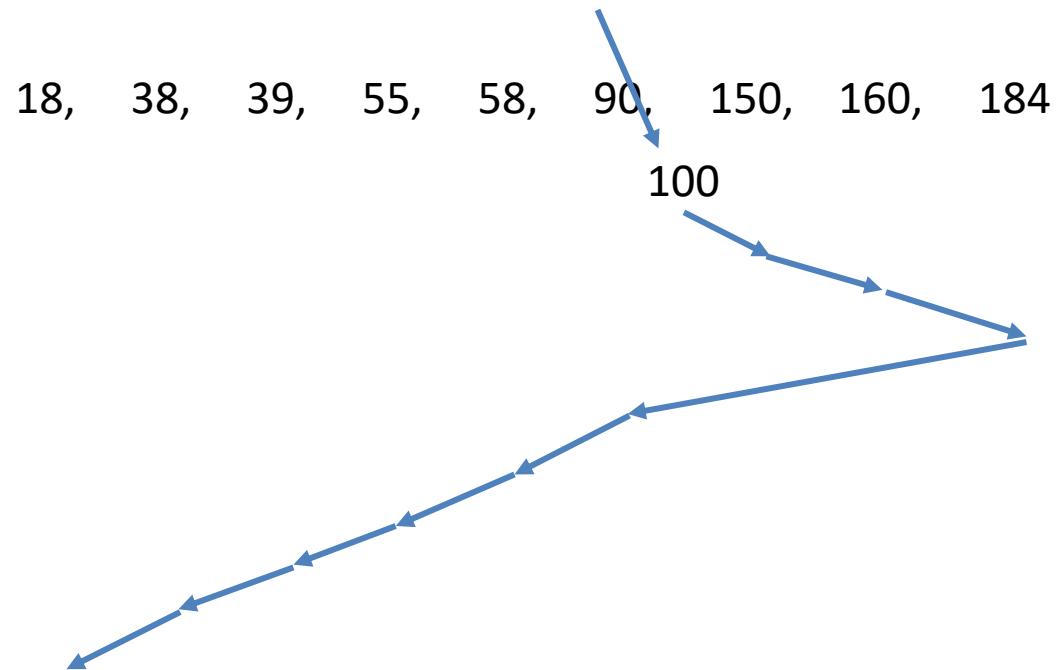
- Scheduling: keep moving in the same direction until there are no more outstanding requests in that direction
- 81, 47, 90, 18, 5;              NOW at 15



# Disk scheduling algorithm: elevator algorithm

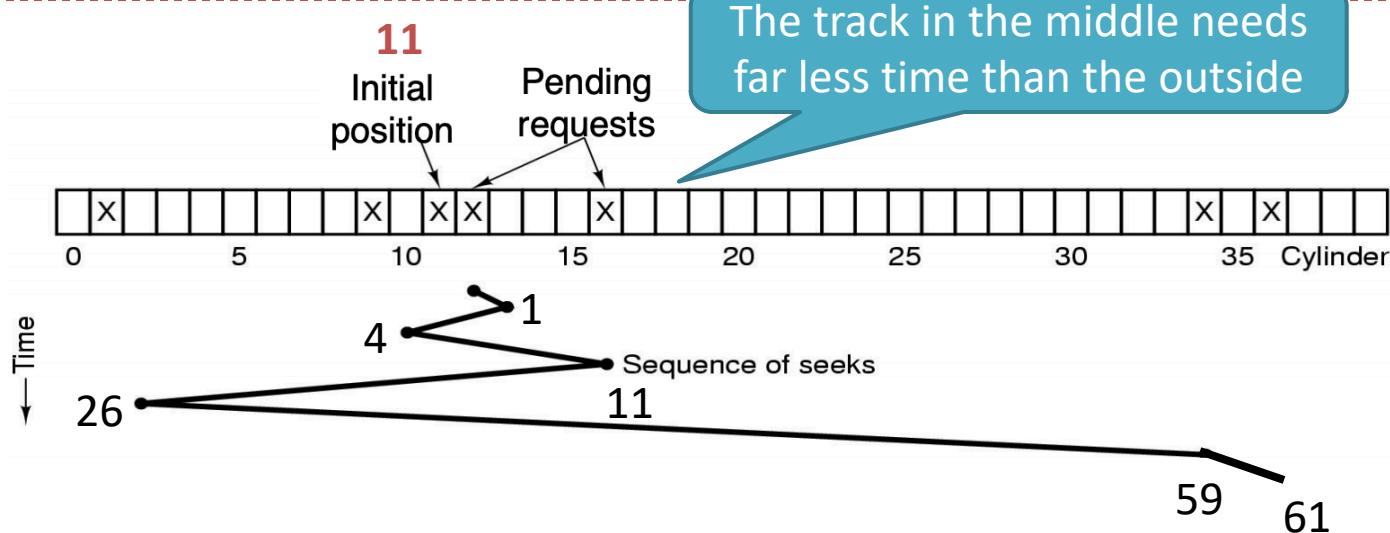
- Example: header start from 100 and the track sequence is 55, 58, 39, 18, 90, 160, 150, 38, 184

Start from 100	
The next track	Distance
150	50
160	10
184	24
90	94
58	32
55	3
39	16
38	1
18	20
Average: 27.8	

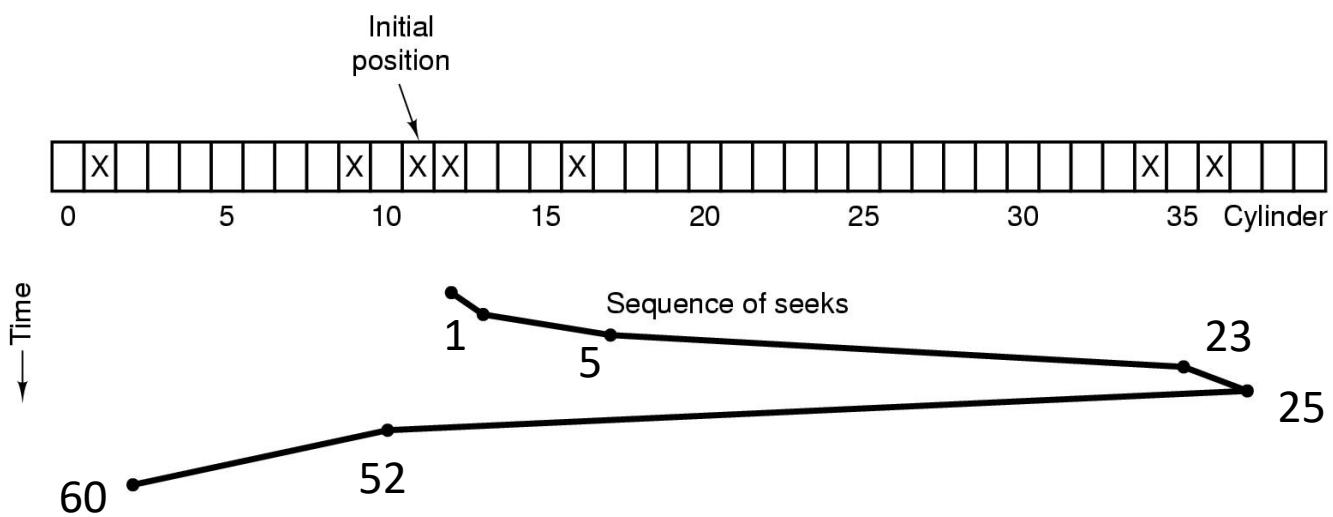


# SSF vs. Elevator algorithm: 1, 36, 16, 34, 9, 12

SSF

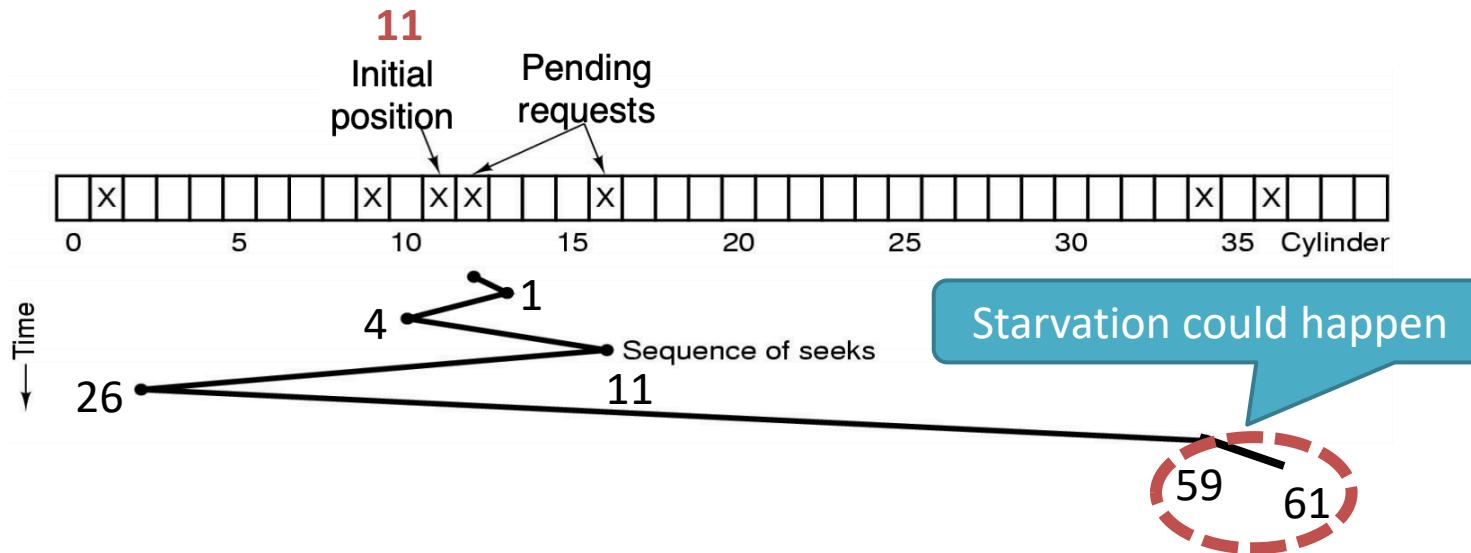


Elevator

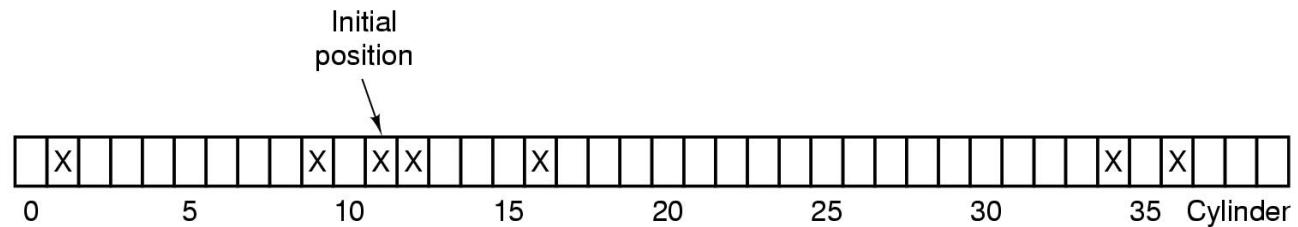


# SSF vs. Elevator algorithm: 1, 36, 16, 34, 9, 12

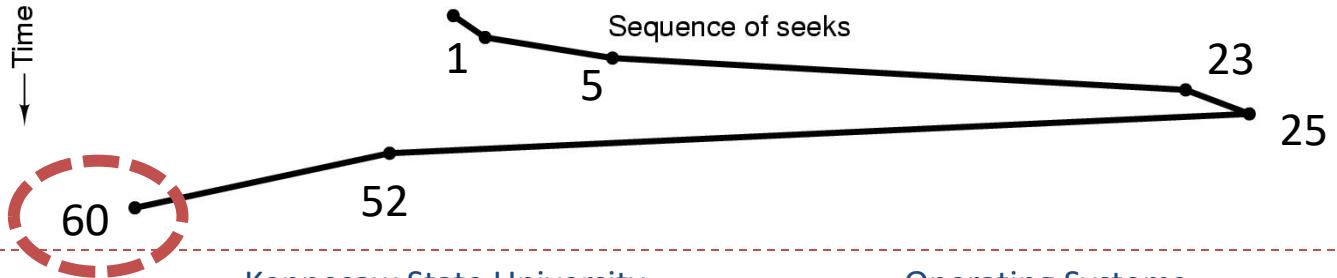
SSF



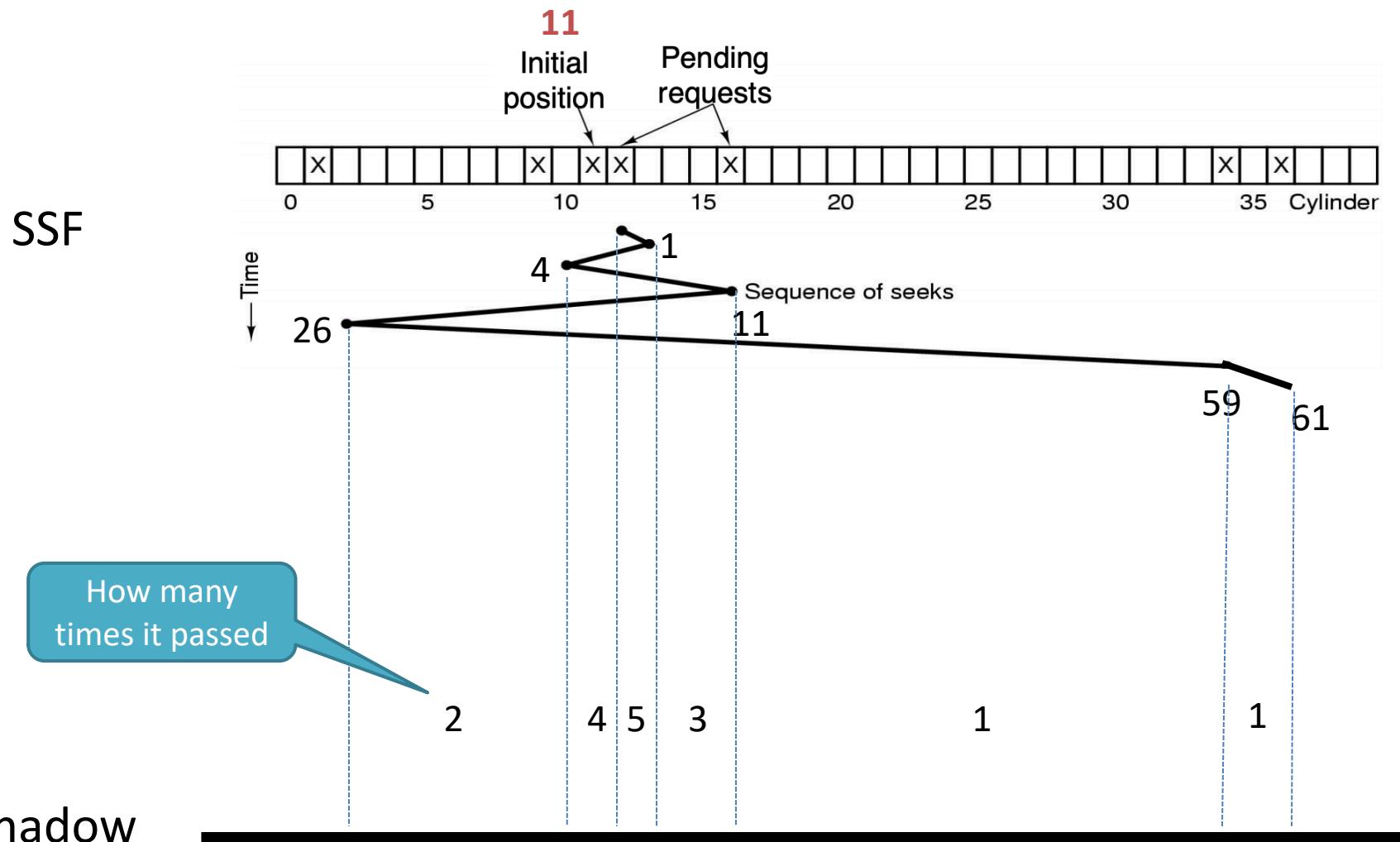
Elevator



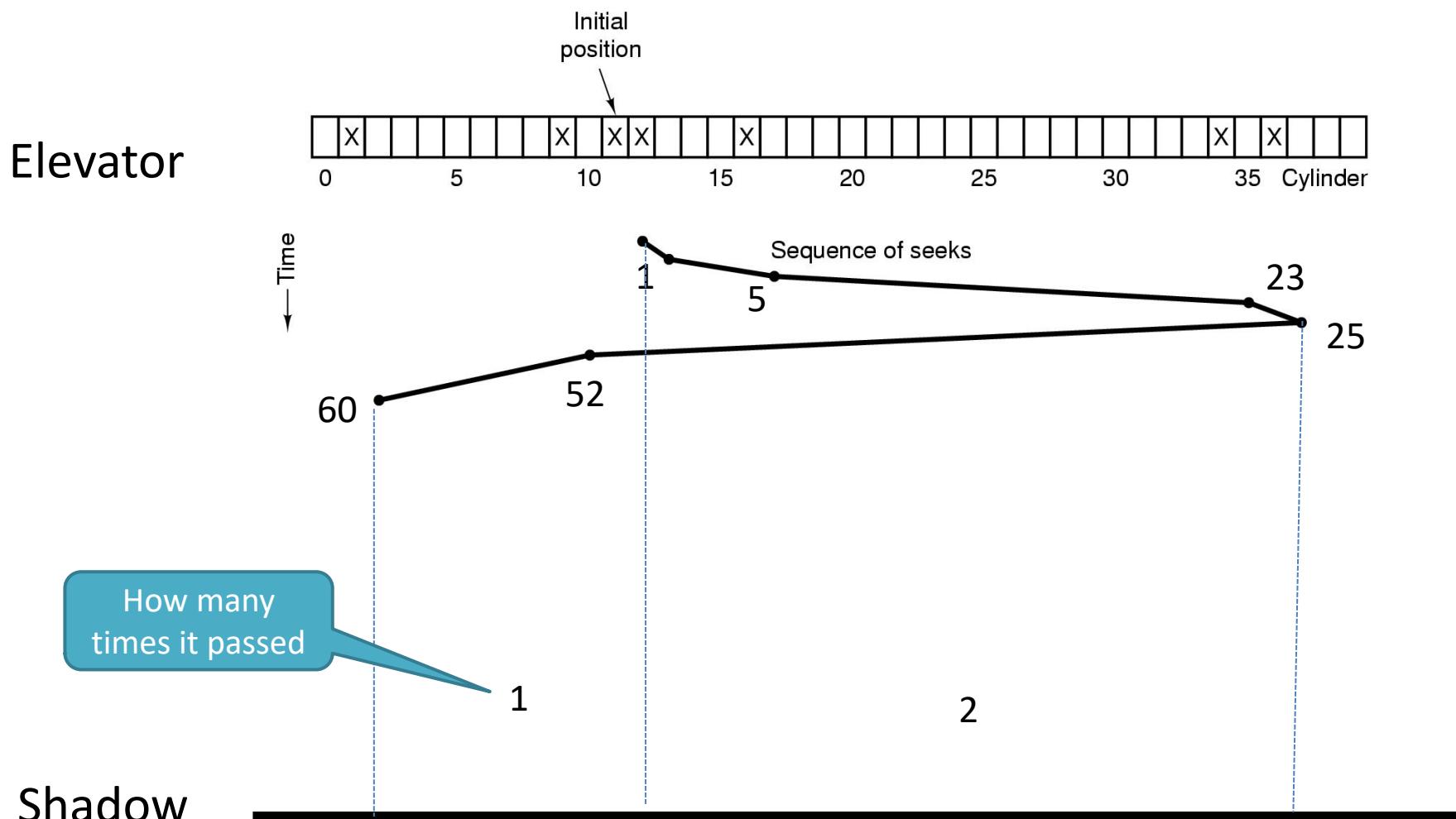
At most  $2 \times \text{length of track}$



# SSF vs. Elevator algorithm: 1, 36, 16, 34, 9, 12



# SSF vs. Elevator algorithm: 1, 36, 16, 34, 9, 12



# Disk scheduling algorithm: elevator algorithm

---

- Pros:
  - Consider both distance and moving direction
  - Avoid starvation (less tracking distance)
- Cons:
  - Certain track might have to wait for long time

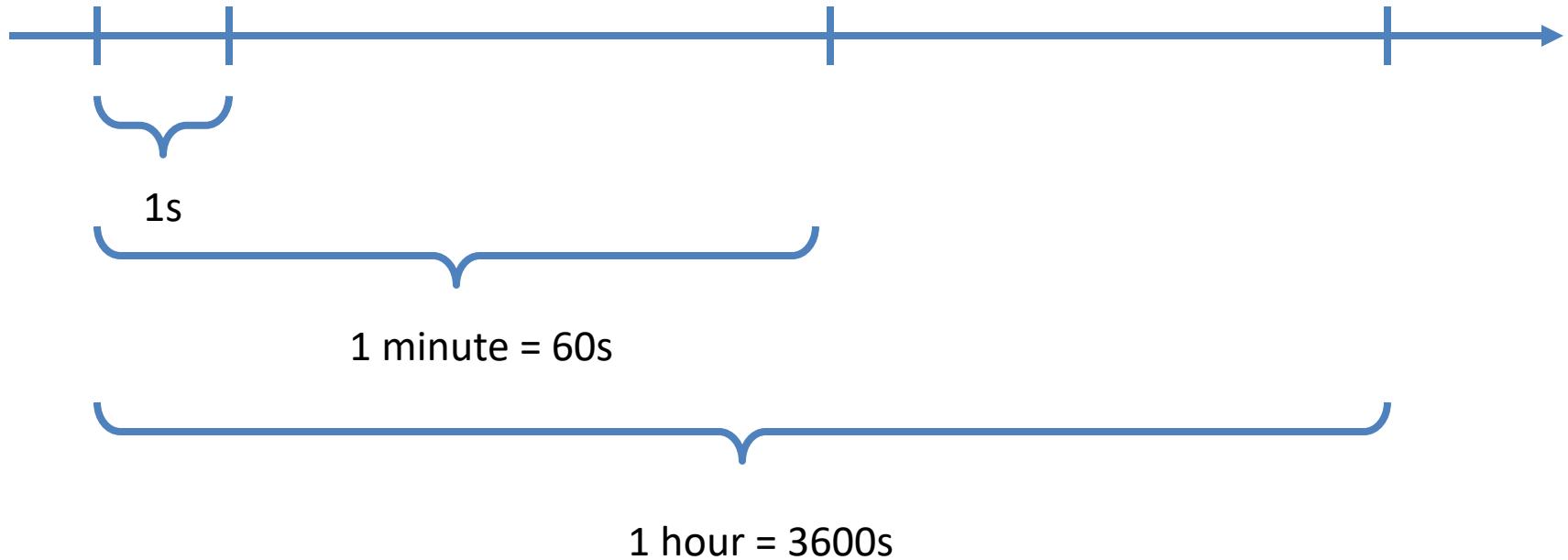
# Outline

---

- Disk scheduling algorithm
- Time in OS and Networking
- Power management in OS
- Virtualization and cloud



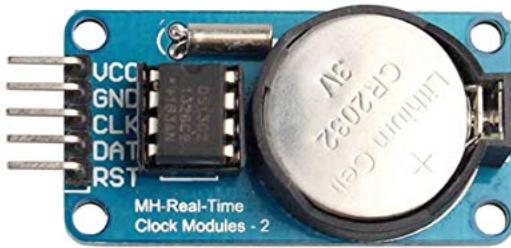
# Time in OS



- Time in the OS is a counter number

# Time in OS: Real Time Clock (RTC)

- The RTC continues to tick even when the machine is powered off, because it is energized by a small battery
- Linux uses the RTC only to derive the time and date



# Time in OS: Time Stamp Counter(TSC)

---

- The counter is accessible through the 64-bit Time Stamp Counter(TSC) register
- Every time the time interrupts come, the TSC counter will be plus by N, (N is related with CPU speed)
- TSC can be read by *native\_read\_tsc()* function
- Example: CPU frequency is 1GHz, every 1ns, the TSC counter is added by 1



# TSC example

```
#include <stdio.h>

typedef unsigned long long u64;
#define DECLARE_ARGS(val, low, high)
#define EAX_EDX_RET(val, low, high)     unsigned low, high
#define EAX_EDX_VAL(val, low, high)    "=a" (low), "=d" (high)
                                         ((low) | ((u64)(high) << 32))

static __always_inline unsigned long long __native_read_tsc(void)
{
    DECLARE_ARGS(val, low, high);
    asm volatile("rdtsc": EAX_EDX_RET(val, low, high));
    return EAX_EDX_VAL(val, low, high);
}

unsigned long long native_read_tsc(void)
{
    return __native_read_tsc();
}

int main()
{
    unsigned long long val, val2;
    unsigned long long delta;

    val = native_read_tsc();
    sleep(1);
    val2 = native_read_tsc();
    delta = val2 - val;

    printf ("v1: %llu, v2: %llu, delta: %llu \n", val, val2, delta);
    printf ("%lld\n", delta/1000000);

    return (0);
}
```

```
[root@disco-0006 ~]# ./test
v1: 6101225874679404, v2: 6101227774910159, delta: 1900230755
1900
```

My CPU is 1.9GHz

$$Time = \frac{t_2 - t_1}{CPU\ frequency}$$

Read two TSC counters  $t_1, t_2$

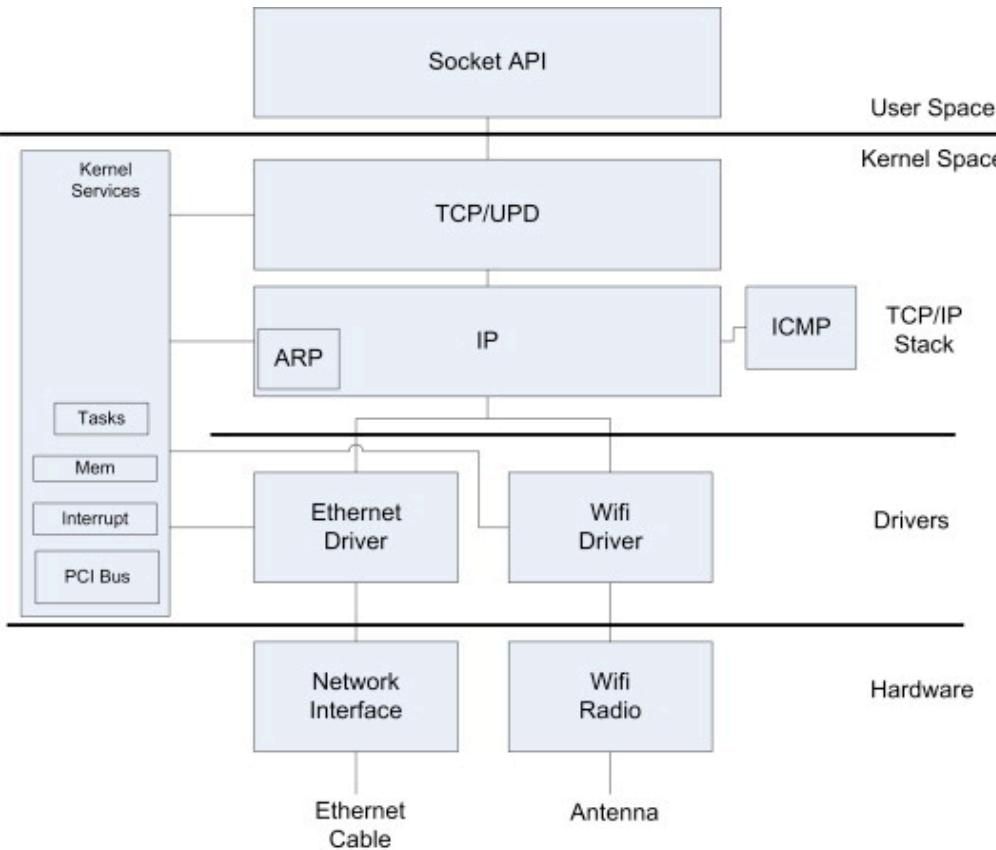
# Other Timers in OS

---

- Programmable Interval Timer (PIT): raise interrupts in a fixed frequency; used for operating system and application programming
- The High Precision Event Timer (HPET): a hardware timer with high precision (nanosecond)
- ACPI power management timer (ACPI PMT): a simple counter that has a fixed frequency clock

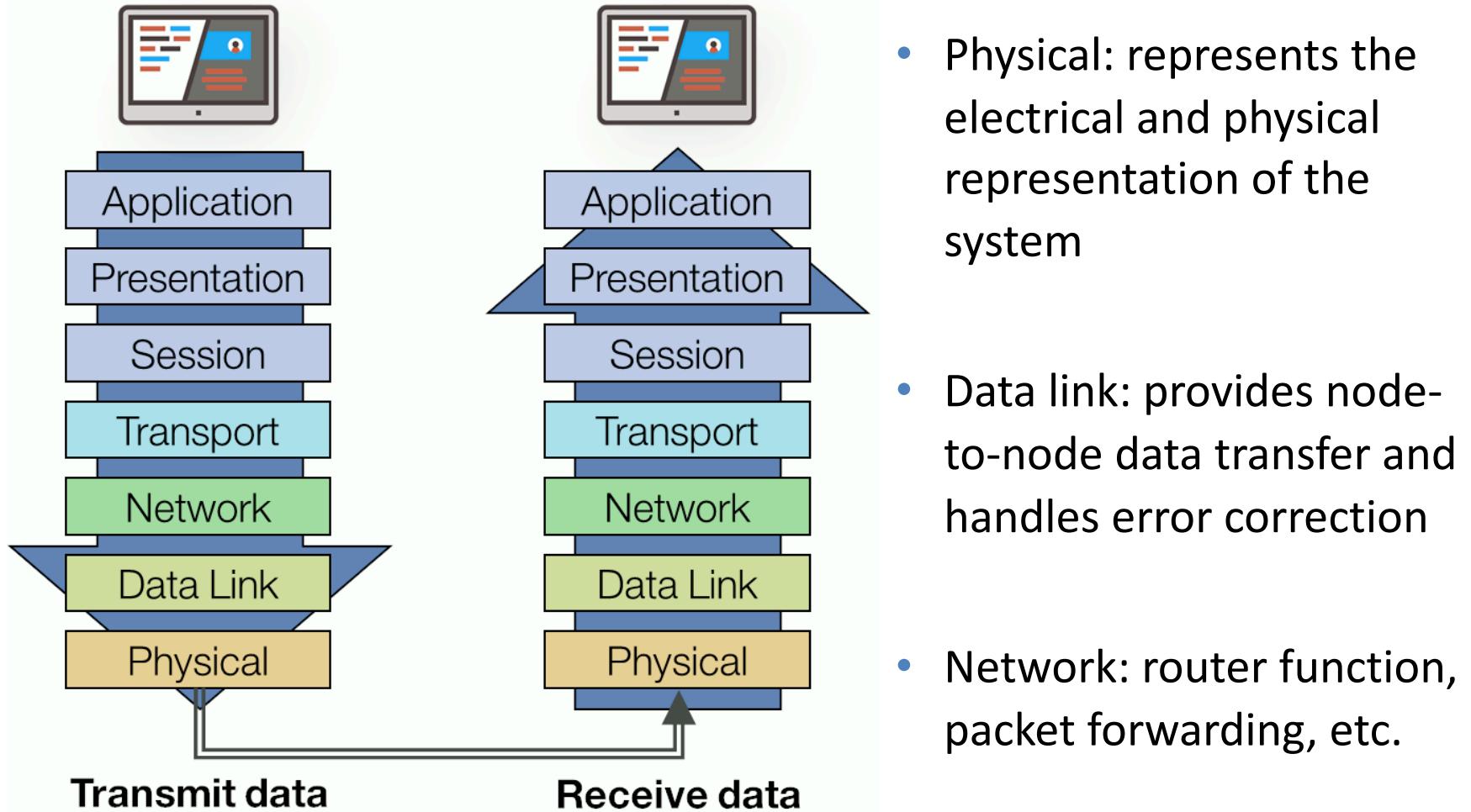


# Networking in OS

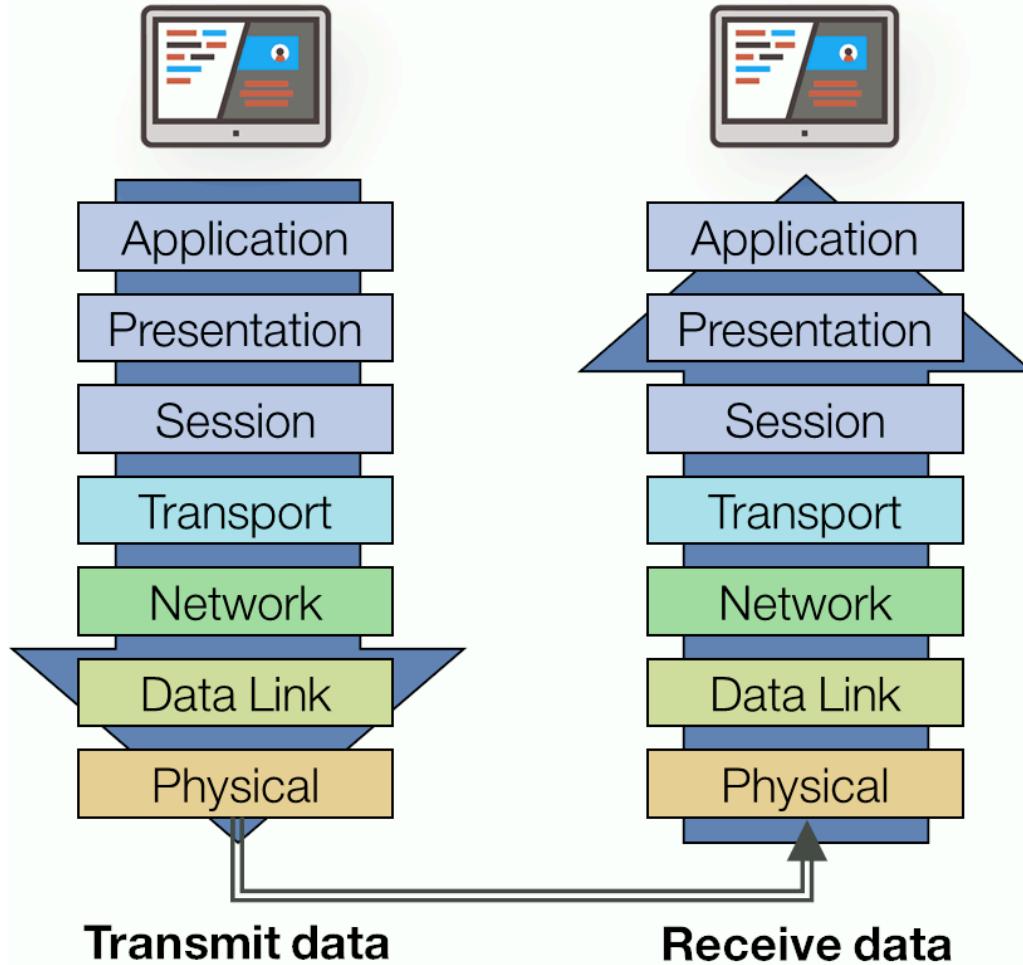


- Network subsystem is part of the I/O system in the OS
- Most operating systems provide a networking stack for the network activities
- Network stack includes IP stack, TCP/UDP protocol, device drivers, socket APIs, etc.

# Networking in OS

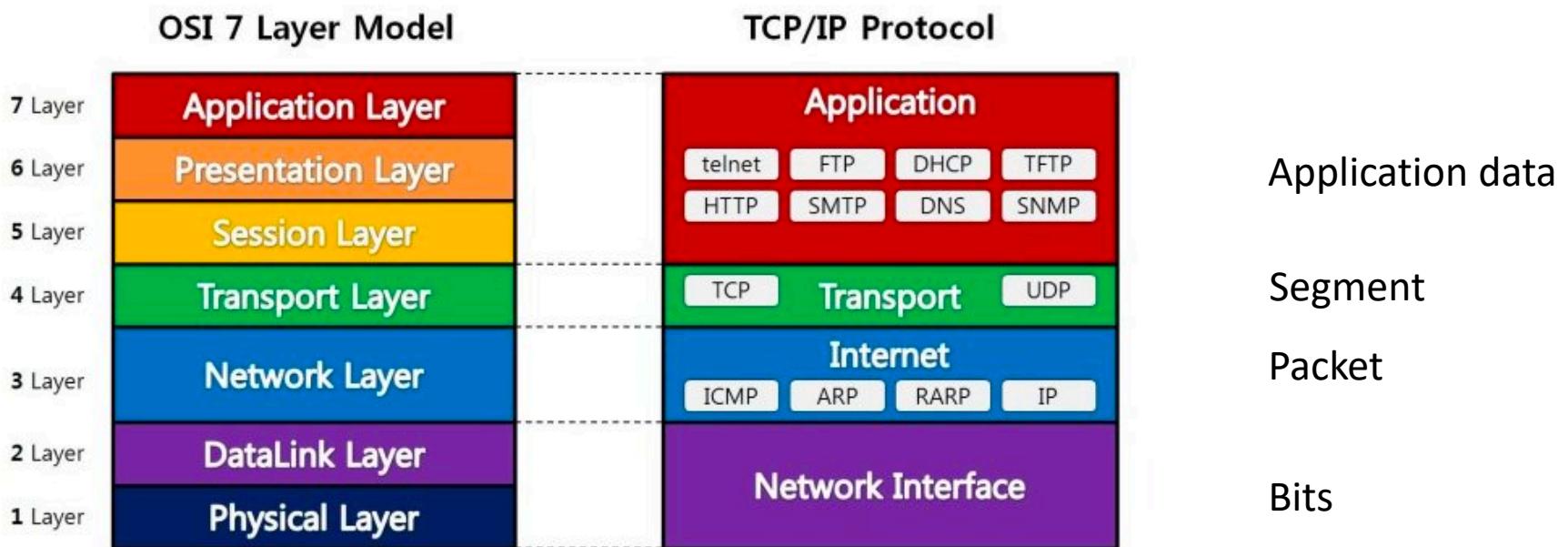


# Networking in OS



- Transport: coordination of the data transfer between end systems and hosts.  
How much data to send, at what rate, where it goes, etc.
- Session/Presentation/Application: interaction with the user level application

# Networking in OS



# Outline

---

- Disk scheduling algorithm
- Time in OS and Networking
- Power management in OS
- Virtualization and cloud



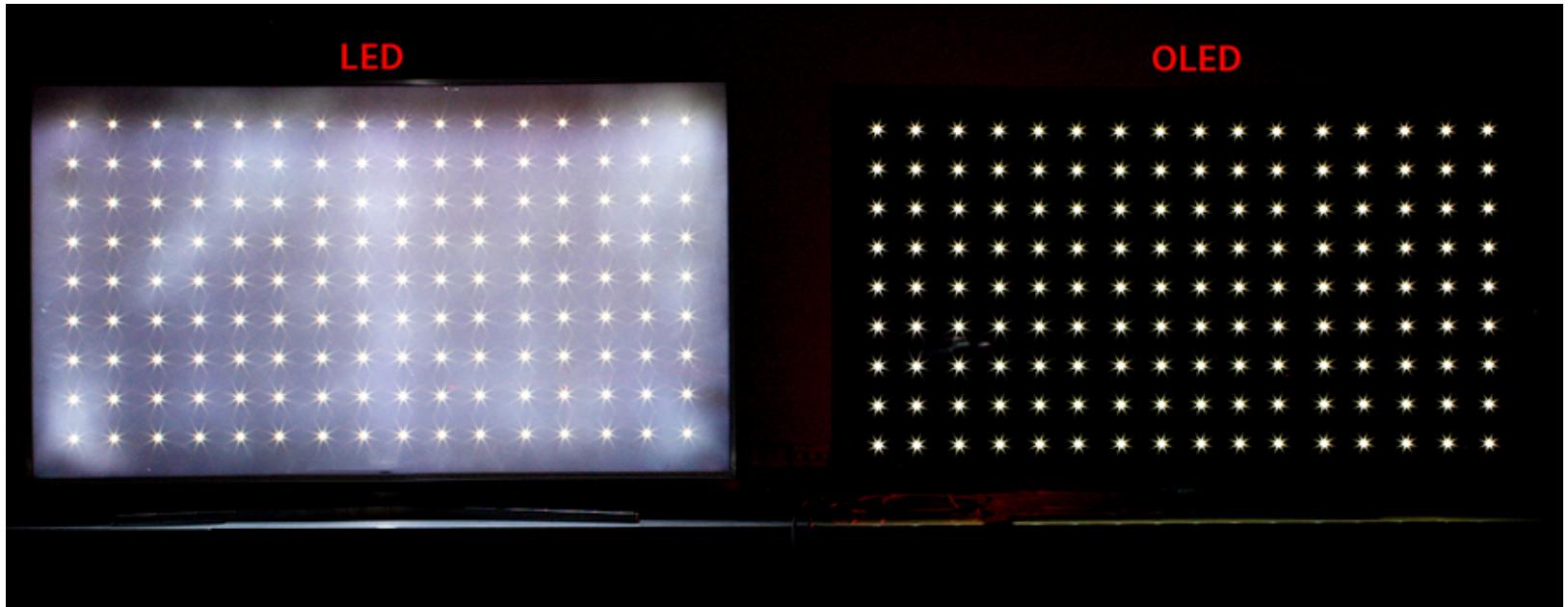
# Power Management

- Power consumption is important to desktops and battery-powered computer
- Two General approaches to battery conservation
  - Turn off (down) some hardware parts, top 3 power eaters: display, hard disk, and CPU
  - Application uses less power, multiple states: on/off, sleep, hibernating

Device	Li et al. (1994)	Lorch and Smith (1998)
Display	68%	39%
CPU	12%	18%
Hard disk	20%	12%
Modem		6%
Sound		2%
Memory	0.5%	1%
Other		22%

Power consumption  
of various parts of a  
laptop computer

# Power Management: The Display



- On-off switching
  - Display consist of some zones that can be independently powered up or down

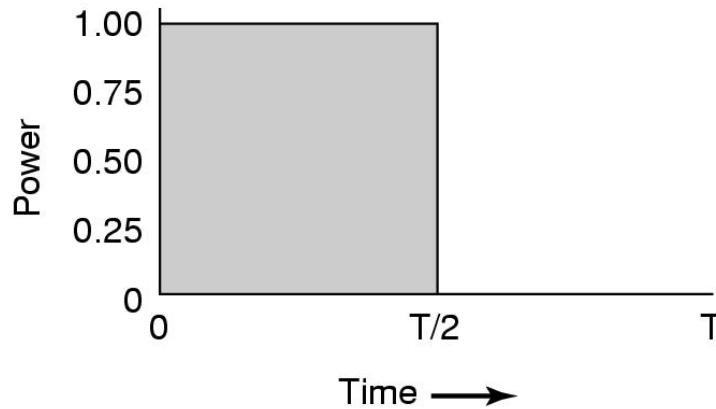
# Power Management: The Hard Disk

---

- It takes substantial energy to keep it spinning at high speed, even if there are no disk accesses
- Restarting a hard disk also consumes considerable energy
- Solution:
  - have a substantial disk cache in RAM

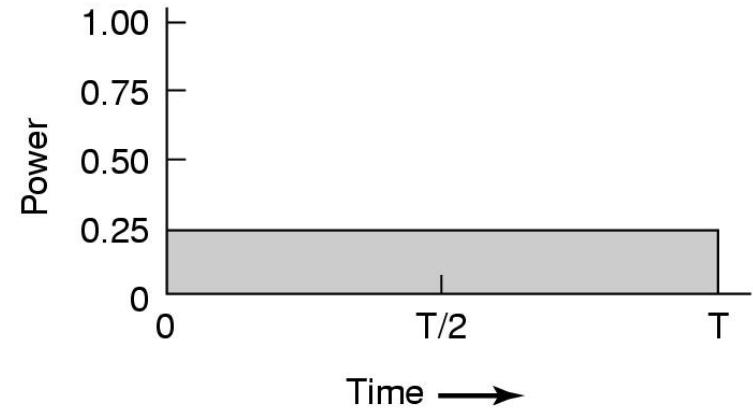


# Power Management: The CPU



Time →

(a)



Time →

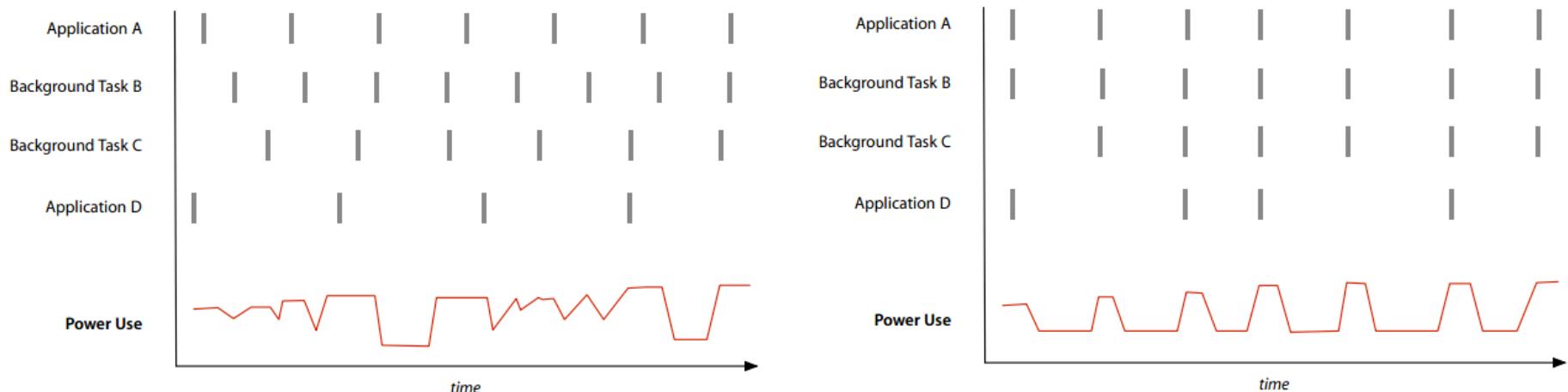
(b)

$$P = CV^2 f$$

- Cut the CPU frequency
- Cut the voltage
- Dynamic voltage and frequency scaling (DVFS): increased or decreased, depending upon circumstances

# Power Management: OS and software techniques

- Timer Coalescing:
- <https://youtu.be/ZIQ5zGOo6qE?t=1652>



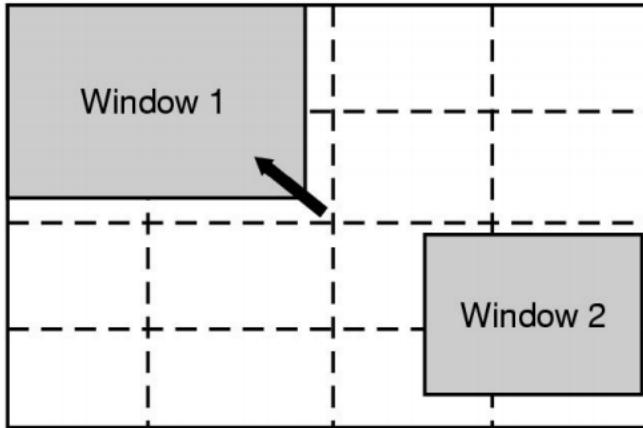
Typically, numerous applications and background processes use timers with different intervals.

Timer Coalescing shifts timers of multiple applications to execute at the same time.



# Power Management: OS and software techniques

- Application nap:
- <https://youtu.be/ZIQ5zGOo6qE?t=1998>



Activity Monitor (Applications in last 12 hours)

App Name	Energy Impact	Avg Energy Impact	App Nap	Requires High Perf G...	Preventing Sle...	User
Google Chrome	36.6	5.56	No	No	Yes	ksuo
Activity Monitor	10.8	0.03	No	No	No	ksuo
Microsoft PowerPoint	2.0	4.20	No	No	No	ksuo
OneDrive	0.6	-	No	No	No	ksuo
Microsoft Outlook	0.6	0.93	No	No	No	ksuo
ESET Endpoint Antivirus	0.6	-	No	No	No	ksuo
Backup & Sync from Google	0.4	0.62	No	No	No	ksuo
ESET Endpoint Antivirus	0.3	-	No	No	No	ksuo
Finder	0.2	0.46	No	No	No	ksuo
Adobe Desktop Service	0.1	-	No	No	No	ksuo
Core Sync	0.1	0.16	No	No	No	ksuo
WhatsApp	0.1	0.21	No	No	No	ksuo
Alertus Desktop	0.1	-	No	No	Yes	ksuo
Grammarly for Safari (Safari)	0.1	-	Yes	No	No	ksuo
欧路词典 鼠标取词	0.1	-	No	No	No	ksuo
Firefox	0.1	0.06	No	No	No	ksuo
Backup & Sync from Google	0.1	-	No	No	No	ksuo
Safari	0.0	0.17	Yes	No	No	ksuo
GitHub Desktop	0.0	-	No	No	No	ksuo
Backup & Sync from Google	0.0	-	No	No	No	ksuo
Backup & Sync from Google	0.0	-	No	No	No	ksuo
Backup & Sync from Google	0.0	-	No	No	No	ksuo

ENERGY IMPACT      Graphics Card: High Perf.  
Remaining charge: 100%  
Battery Is Charged  
Time on AC: 1:43

BATTERY (Last 12 hours)



# Outline

---

- Disk scheduling algorithm
- Time in OS and Networking
- Power management in OS
- Virtualization and cloud



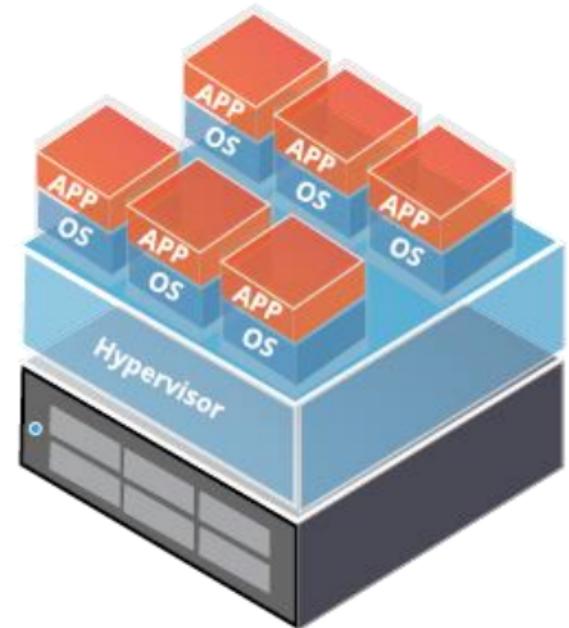
# How to share a physical computer

- Operating systems allows multiple processes/applications to run simultaneously
- Via process/memory management, CPU scheduler, multithreading, concurrency control
- Allow multiple users to access the same machines



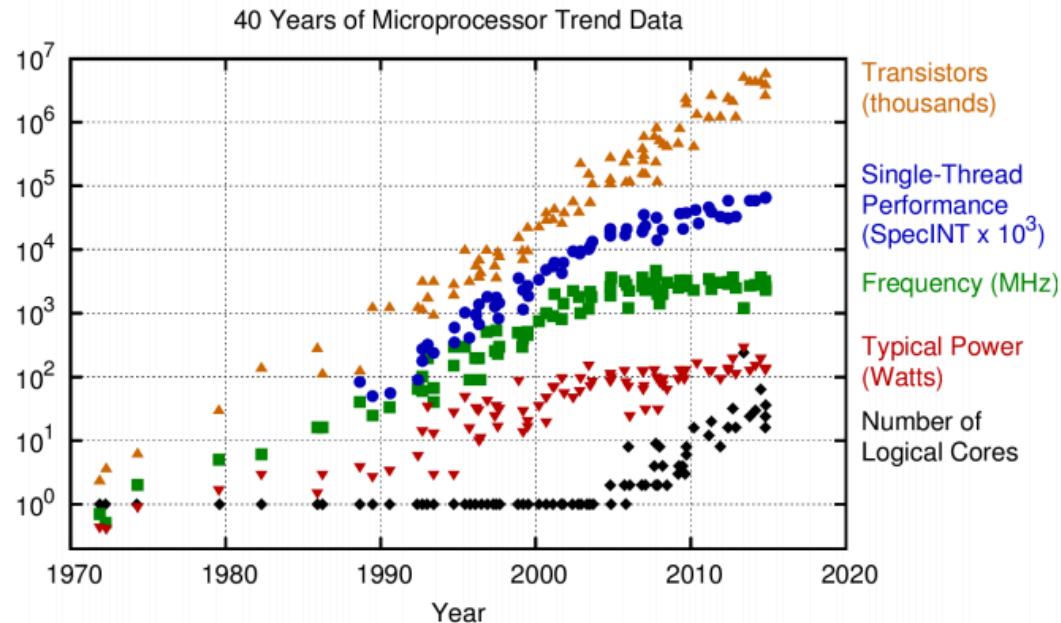
# Virtual Machines

- Run multiple (virtual) machines on the same server to share the same physical resources, including CPU, memory, storage, etc.
- Provide the illusion to the virtual machines that they exclusively occupy the physical resources



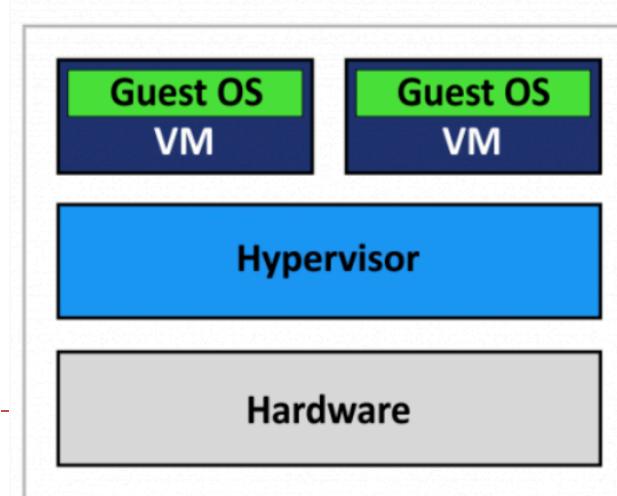
# Why VM?

- Single core becomes more powerful
- More and more cores per CPU
- Overall CPU% is low (in data centers) – 15% ~ 18%
- How to efficiently use them? -- Sharing

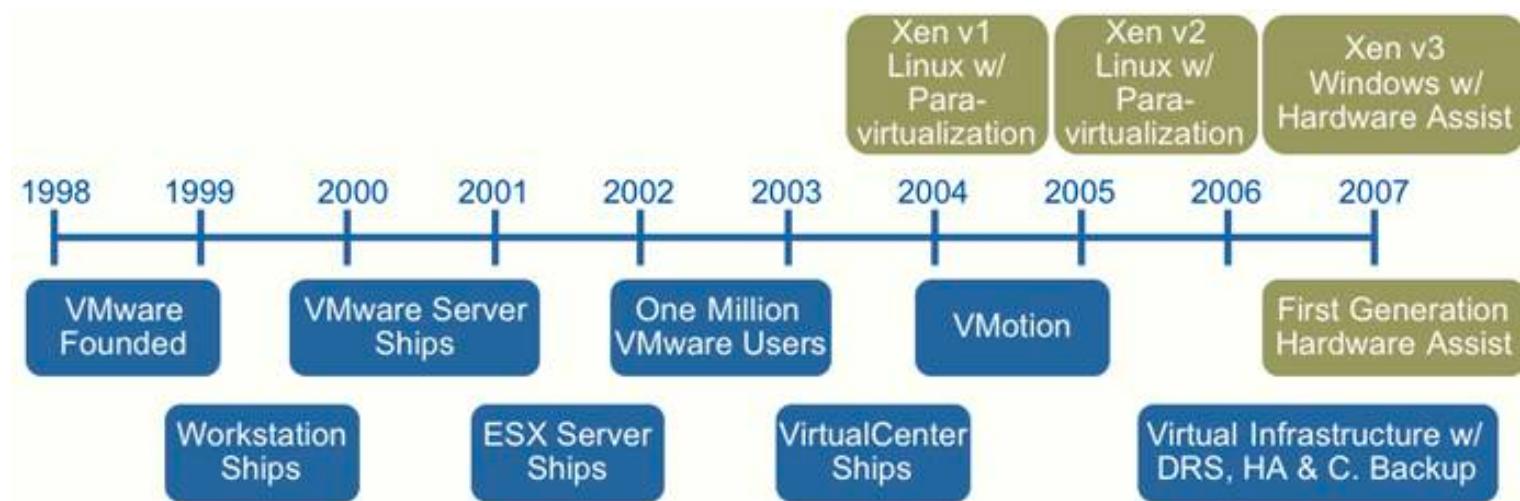


# Solutions for hypervisor

- Commercial virtual machines for x86 architecture
  - VMware (1999-)
  - Connectix Virtual PC (now Microsoft)
  - KVM for Linux
- Research virtual machines for x86 architecture
  - Xen (SOSP '03)
  - plex86

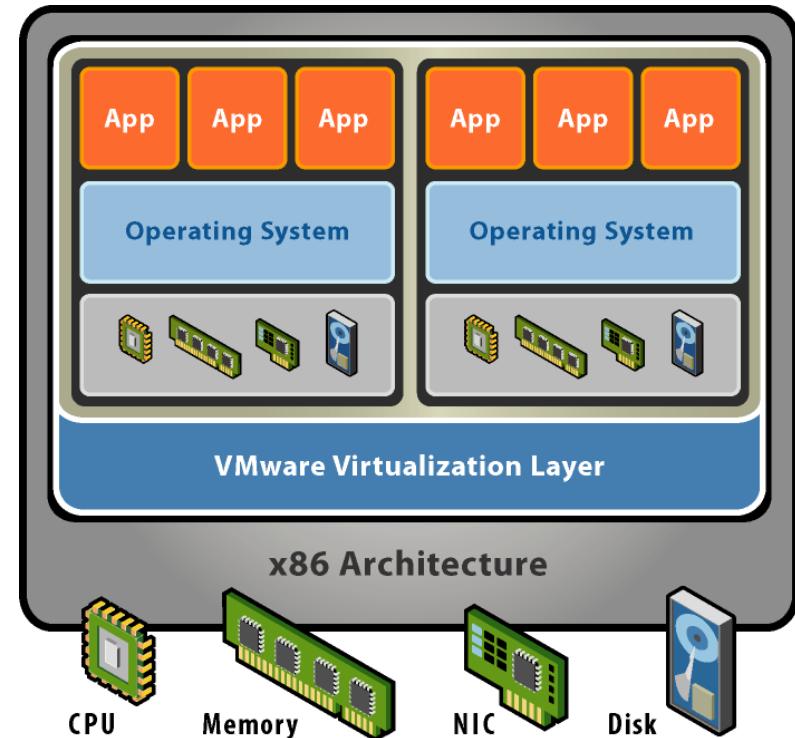


# Virtualization history



# Resource Virtualization

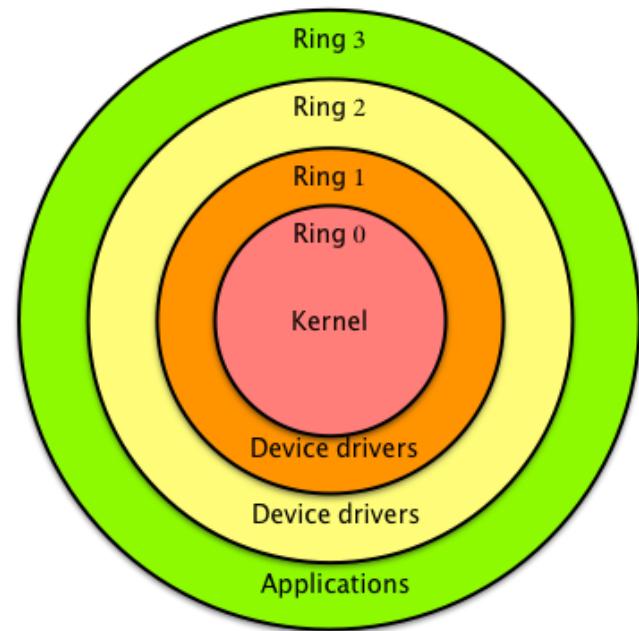
- Processor
- Memory
- Device and I/O



# CPU Rings

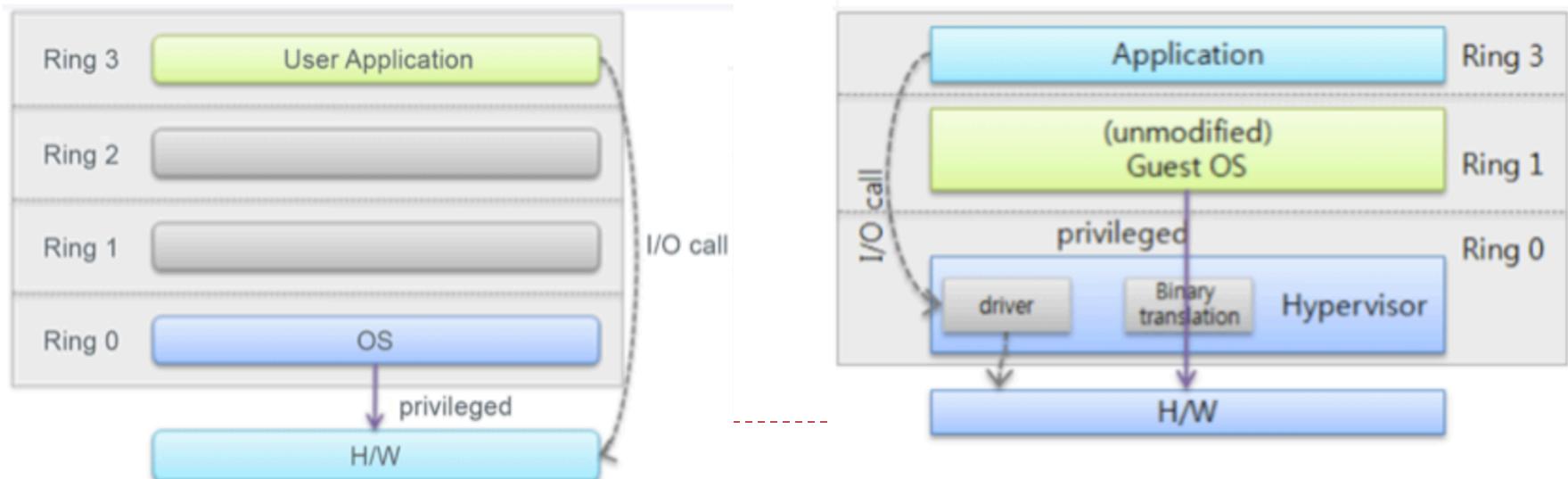
---

- User and kernel mode are controlled by CPU
- Multiple CPU protection rings
  - traditional OS runs in ring 0
  - Application runs in ring 3
  - must handle ring 3 to ring 0 transition



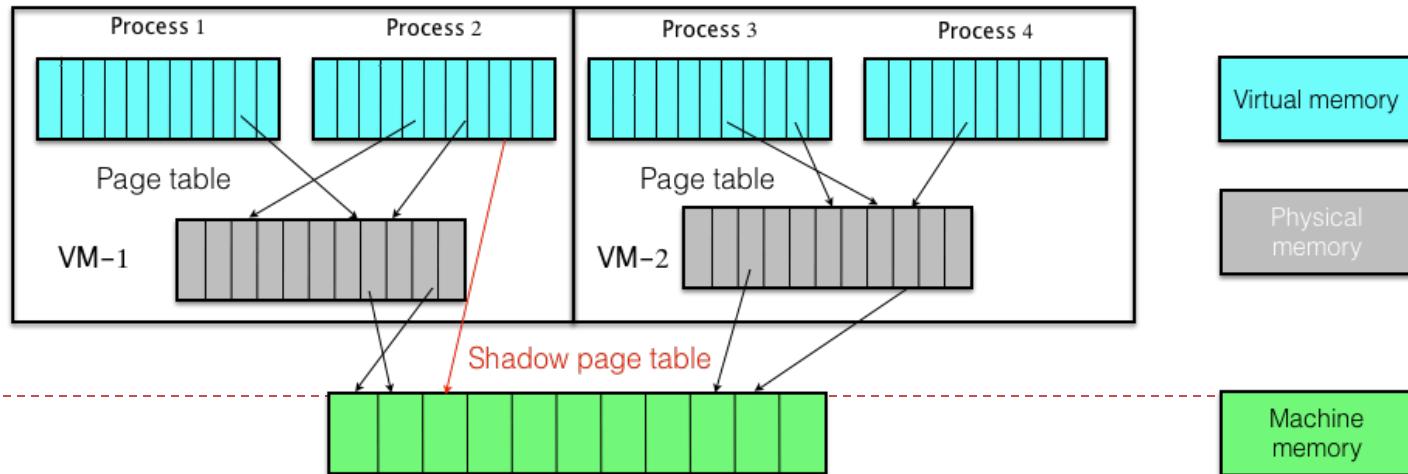
# CPU Virtualization

- Place Guest OS in Ring 1, and VMM in Ring 0 (with the highest privilege)
- if Guest OS executes privilege instructions:
  - Cause a trap if executed in privilege level other than Ring 0
  - VMM/hypervisor interprets such instructions and do something



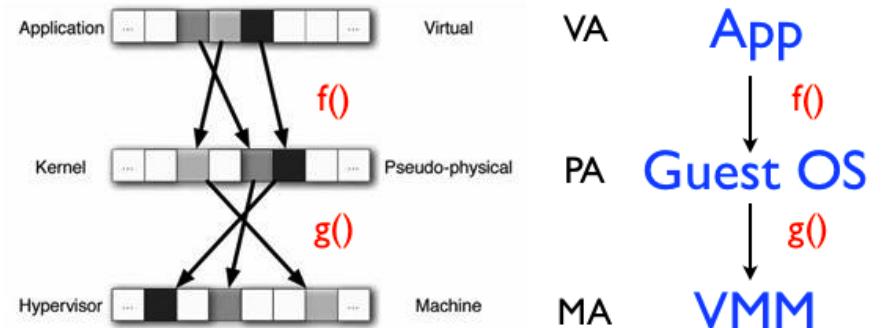
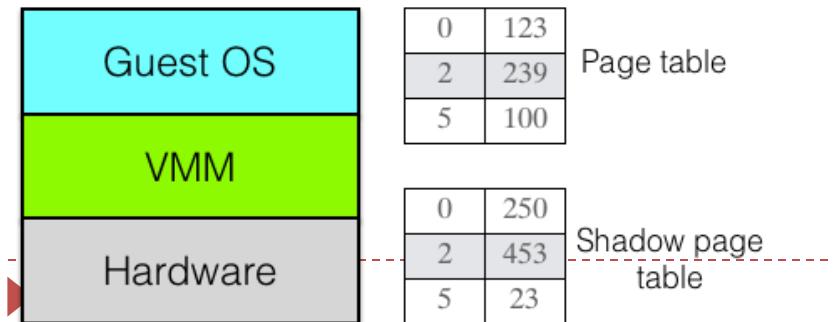
# Memory Virtualization

- Three memory addresses
  - virtual memory (process), physical memory (OS), machine memory (VMM)
  - VMM maintains a shadow mapping from VA to MA



# Software MMU

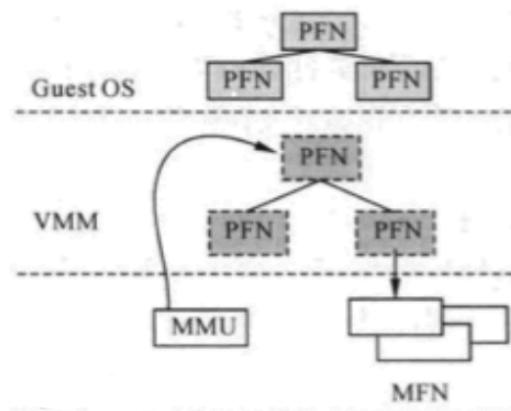
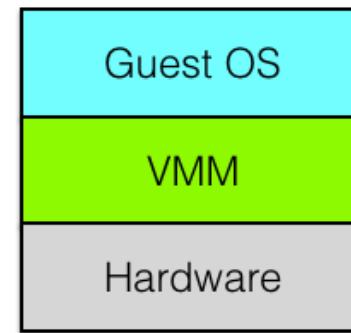
- High virtualization overhead with shadow page table
  - frequent guest OS to VMM transition and TLB flush
  - Xen's optimization
    - ▶ directly register guest PG to MMU
    - ▶ Read to PG bypass VMM
    - ▶ VMM traps updates to VMM
    - ▶ Batch updates
    - ▶ Reserve top 64MB for VMM to avoid TLB flush due to guest/VMM switch



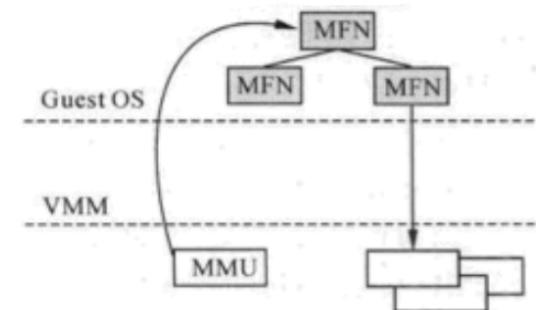
# Hardware assist MMU

- Extended/Nested page tables

- Intel VT-x and AMD-V
- no shadow page table is needed
- Two hardware PGs
  - ▶ VA->PA and PA->MA
- Tagged TLB entry
- costly page walk



Shadow page table

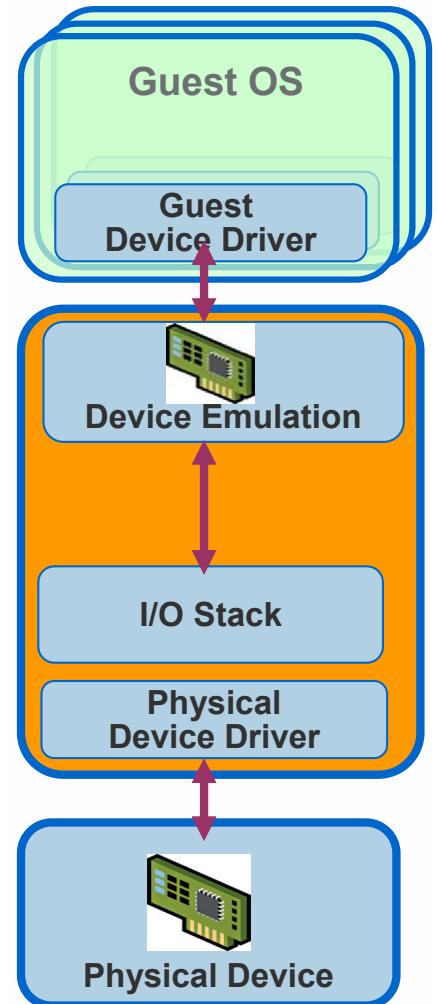


Non-Shadow page table

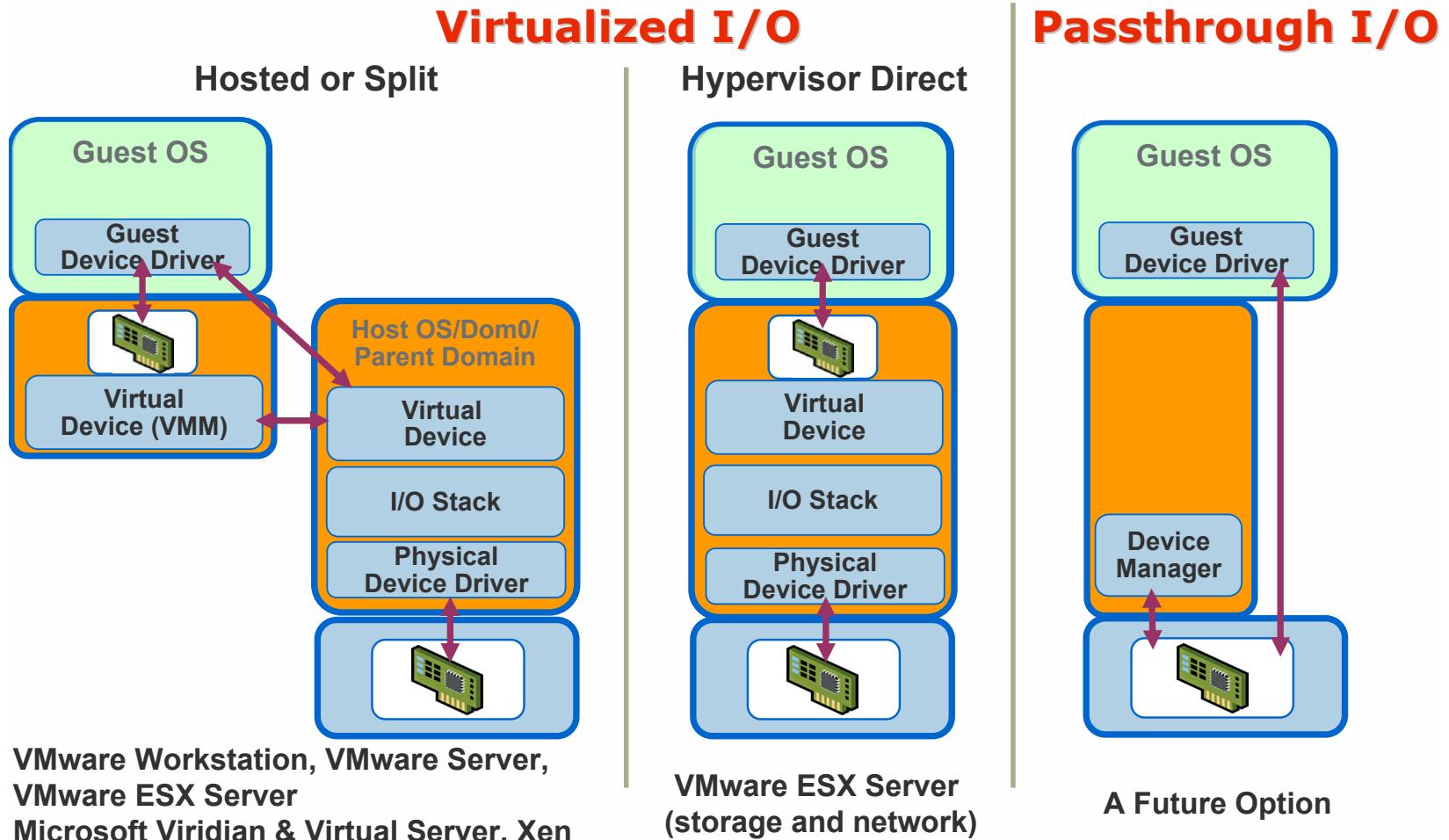


# I/O Virtualization

- I/O virtualization architecture
  - guest driver
  - generic virtual device, e.g., Intel e1000
  - virtualization I/O stack
  - real device driver
  - hardware device

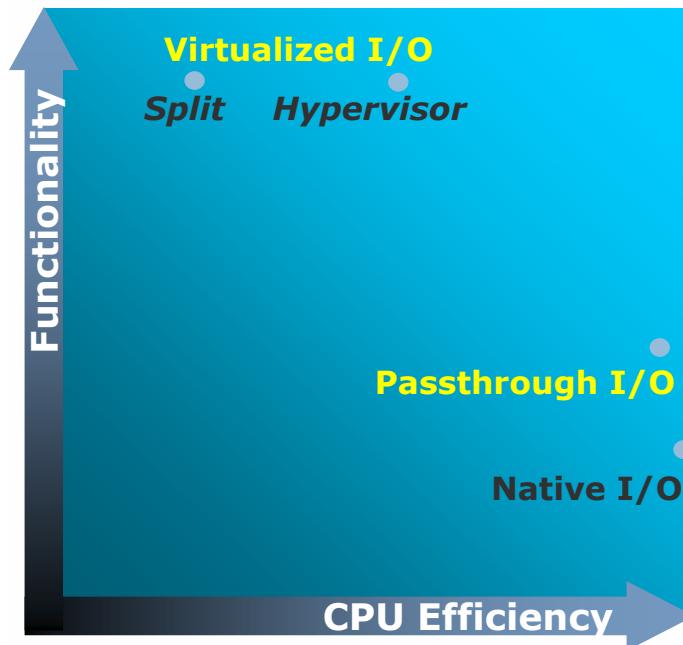


# I/O Virtualization Implementations



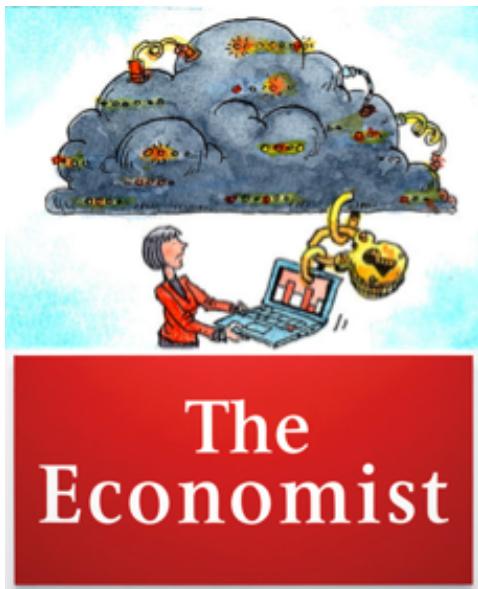
# I/O Virtualization Trade-offs

- Virtualized I/O provides rich functionality
- Passthrough I/O reduces CPU utilization and better performance



# The Rise of Cloud Computing

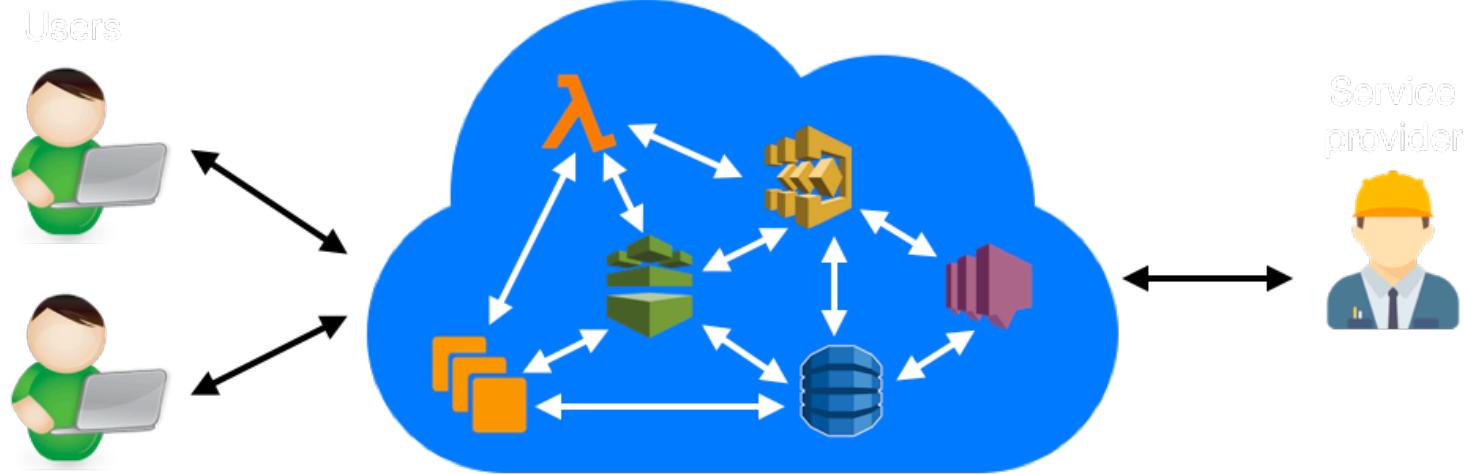
---



The rise of the cloud is more than just another platform shift that gets geeks excited. It will undoubtedly transform the information technology (IT) industry, but it will also profoundly change the way people work and companies operate.

— A survey of corporate IT – Let it rise  
By the Economist  
Oct 23rd 2008

# The Benefits of Clouds



Easy to use  
High performance  
Stable user experience  
Automatic software updates  
...

Easy to manage  
High resource utilization  
Low operation cost  
Energy efficiency  
...

# Virtualization — the Rock of Cloud Computing

	Level	Example	Performance	Complexity	Isolation
App	Application level	JVM, .Net	xx	xxxxx	xxxxx
Library	Library level	WINE, vCUDA	xxx	xx	xx
Container	OS level	Docker, LXC	xxxxx	xxx	xx
VM	Hardware level	Vmware, Xen, KVM	xxxxx	xxxxx	xxxx
Hardware	Instruction level	QEMU, Bochs	x	xxx	xxx



# More services on the cloud

2006



1st AWS Service  
S3



Now: 3149 New & Existing Services

The screenshot shows the AWS service catalog interface. At the top, it says "Amazon Web Services" and "aws.amazon.com". Below that, there are several sections of services:

- Compute**: EC2 (Virtual Servers in the Cloud), EC2 Container Service (Run and Manage Docker Containers), Elastic Beanstalk (Run and Manage Web Apps), Lambda (Run Code in Response to Events).
- Storage & Content Delivery**: S3 (Scalable Storage in the Cloud), CloudFront (Global Content Delivery Network), Elastic File System (Fully Managed File System for EC2), Glacier (Archive Storage in the Cloud), Import/Export Snowball (Large Scale Data Transport), Storage Gateway (Integrates On-Premises IT Environments with Cloud Storage).
- Database**: RDS (Managed Relational Database Service), DynamoDB (Predictable and Scalable NoSQL Data Store), ElastiCache (In-Memory Cache), Redshift (Managed Petabyte-Scale Data Warehouse Service).
- Networking**: VPC (Isolated Cloud Resources), Direct Connect (Dedicated Network Connection to AWS), Route 53 (Resolve DNS and Domain Name Registration).
- Developer Tools**: CodeCommit (Store Code in Private Git Repositories), CodeDeploy (Automate Code Deployments), CodePipeline (Release Software using Continuous Delivery).
- Management Tools**: CloudWatch (Monitor Resources and Applications), CloudFormation (Create and Manage Resources with Templates), CloudTrail (Track User Activity and API Usage), Config (Track Resource Inventory and Changes), OpsWorks (Automate Operations with Chef), Service Catalog (Create and Use Standardized Products), Trusted Advisor (Optimize Performance and Security).
- Security & Identity**: Identity & Access Management (Manage User Access and Encryption Keys), Directory Service (Host and Manage Active Directory), Inspector (Analytics Application Security), WAF (Filter Malicious Web Traffic).
- Analytics**: EMR (Managed Hadoop Framework), Data Pipeline (Orchestration for Data-Driven Workflows), Elasticsearch Service (Run and Scale Elasticsearch Clusters).
- Internet of Things**: AWS IoT (Connect Devices to the cloud).
- Mobile Services**: Mobile Hub (Build, Test, and Monitor Mobile apps), Cognito (User Identity and App Data Synchronization), Device Farm (Test Android, Fire OS, and iOS apps on real devices in the Cloud), Mobile Analytics (Collect, View and Export App Analytics), SNS (Push Notification Service).
- Application Services**: API Gateway (Build, Deploy and Manage APIs), AppStream (Low Latency Application Streaming), CloudSearch (Managed Search Service), Elastic Transcoder (Easy-to-use Scalable Media Transcoding), SES (Email Sending Service), SQS (Message Queue Service), SWF (Workflow Service for Coordinating Application Components).
- Enterprise Applications**: WorkSpaces (Desktops in the Cloud), WorkDocs (Secure Enterprise Storage and Sharing Service), WorkMail (Secure Email and Calendaring Service).



# Current cloud ecosystem

Cloud Native Landscape v20180425 See the interactive landscape at [l.cnfc.io](https://l.cnfc.io) Greyed logos are not open source

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

App Definition and Development

- Database and Data Warehouse
- Streaming
- Source Code Management
- Application Definition and Image Build
- Continuous Integration / Continuous Delivery (CI/CD)

Orchestration & Management

- Scheduling & Orchestration
- Coordination & Service Discovery
- Service Management

Runtime

- Cloud-Native Storage
- Container Runtime
- Cloud-Native Network

Provisioning

- Host Management / Tooling
- Infrastructure Automation
- Container Registries
- Secure Images
- Key Management

Cloud

- Public
- Private

Platforms

- Certified Kubernetes - Distribution
- Certified Kubernetes - Hosted
- Certified Kubernetes - Installer
- Non-Certified Kubernetes

Observability & Analysis

- Monitoring
- Logging
- Tracing
- Serverless
- Kubernetes Training Partner

The Cloud Native Landscape Map displays a wide range of open-source and commercial projects organized into various categories:

- App Definition and Development:** Database and Data Warehouse (e.g., Vitess, Apache Cassandra, MongoDB), Streaming (Apache Flink, Apache Nifi), Source Code Management (GitHub, GitLab), Application Definition and Image Build (Dockerfile, Packer), Continuous Integration / Continuous Delivery (CI/CD) (Jenkins, CircleCI, Travis CI).
- Orchestration & Management:** Scheduling & Orchestration (Kubernetes, CoreOS), Coordination & Service Discovery (CoreDNS, Consul, etcd), Service Management (Envoy, Linkerd, Open Policy Agent).
- Runtime:** Cloud-Native Storage (ROCK, Ceph, CSI, Gluster, Hedvig), Container Runtime (Containerd, rkt, cri-o, kata, Podman), Cloud-Native Network (CNI, Aposesto, Haproxy, Envoy, Openvswitch, Opencontrail, Cilium, NSX).
- Provisioning:** Host Management / Tooling (Ansible, CFEngine, Chef), Infrastructure Automation (BOSH, Juju, Terraform), Container Registries (Docker, Kubernetes, Quay, Artifactory), Secure Images (Spiffe, Spire), Key Management (Pivotal, Conjur, Vault).
- Cloud:** Public Cloud Providers (AWS, Azure, Google Cloud, IBM Cloud, Oracle, etc.) and Private Cloud Options (VMware, MAAS, openstack).
- Platforms:** Certified Kubernetes - Distribution (Amazon EKS, Docker, Kubernetes, OpenShift), Certified Kubernetes - Hosted (OpenShift, Kubernetes), Certified Kubernetes - Installer (CoreOS, Google Cloud, Heptio), Non-Certified Kubernetes (Tectonic, RKE, Rancher, K3s, EFK).
- Observability & Analysis:** Monitoring (Prometheus, Grafana, InfluxDB), Logging (Logstash, Logstash, Logstash), Tracing (Jaeger, OpenTelemetry, Zipkin), Serverless (Flynn, OpenFaaS, Serverless), Kubernetes Training Partner (various providers).

Greyed logos indicate they are not open source projects.

# Conclusion

---

- Disk scheduling algorithm
- Time in OS and Networking
- Power management in OS
- Virtualization and cloud

