DISTRIBUTED AND CLOUD COMPUTING

Dr. Padraig Corcoran

Course overview

- Lecturer is Dr. Padraig Corcoran.
- Sessions consisting of lecture and lab:
  - Lecture - Tuesday 9:00-10:50 (lectures are recorded).
  - Lab - Tuesday 13:10-15:00.
  - Week 3 - no lecture.
- Assessment:
  - Coursework 30% (two 15% assignments)
    - Coursework 1 - hand out wk 3, hand in wk 6, feedback wk 9.
    - Coursework 2 - hand out wk 7, hand in wk 10, feedback wk 12.
  - Exam 70%.

Text books

- Van Steen and Tanenbaum. Distributed Systems. Pearson. 2017. (www.distributed-systems.net).
- Ghosh. Distributed Systems: an Algorithmic Approach (Second Edition). CRC press. 2014.
- Vacca. Cloud Computing Security: Foundations and Challenges. CRC Press. 2016.

Course content

Lecture topics	Lab topics
Introduction	Python programming (wk 1)
MapReduce	MapReduce (wk 2)*
System architectures	Python support (wk 3)
Processes and threads	Distribution transparency (wk 4)
Communication	Virtualization (wk 5)
Web services	Remote objects (wk 6)*
Coordination	Processes & threads (wk 7)
Security	Network analysis (wk 8)
	Web services (wk 9)
	Secure communication (wk 10)
	Digital signatures (wk 11)

* coursework dependant

What is a distributed system?

- Computer networks define connectivity between computers.
- The services build on top of these networks are examples of distributed systems.
- Van Steen (2017) states:

"A distributed system is a *collection* of *autonomous computing elements* that *appears to its users as a single coherent system*"

- Ghosh (2014) states that a distributed system typically satisfies four criteria:
  - Multiple processes – more than one sequential process. Each process should have an independent thread of control.
  - Interprocess communication – processes communication using *messages*. Message links are known as *channels*.
  - Disjoint address spaces – Process have *disjoint address spaces*. A shared memory multiprocessor is not considered a true representative of a distributed system.
  - Collective goal – processes must interact to meet a common goal.

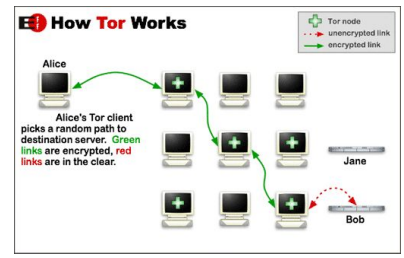
Why distributed systems?

Reasons for the growing importance of distributed systems:

- Resource sharing - this includes hardware (e.g. printer, computation power), software (e.g. Google Docs) and information.
- Geographical distributed environments - in many situations, the computing environment itself is geographically distributed (e.g. banking, the WWW).
- Speed up - the speed of computation in traditional uniprocessors is fast approaching the physical limit. More computational power can be achieved using multiple processors.
- Fault tolerance - when a fraction of the processors fail, the remaining processors keep the application running. Allows for *graceful degradation*.

Tor (The Onion Router)

Tor is free software for enabling anonymous communication.



Large Hadron Collider (LHC)

- LHC is a particle accelerator and complex experimental testbed build at the European Organisation for Nuclear Research (CERN).
- Aims to answer fundamental scientific questions.
- Data stored in a distributed manner.

CERN's Large Hadron Collider

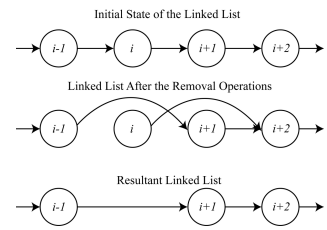
SETI@home project

- SETI (search for extraterrestrial intelligence) aims to discover the existence of extraterrestrial life in the universe.
- Large volumes of data is constantly collected from hundreds of radio telescopes.
- Data processed in a distributed manner.

Radio telescopes

Mutual exclusion

- Access to certain hardware and software must be restricted to one process at a time.
- Mutual exclusion guarantees that at most one process acquires the resource or performs a critical operation at any time.
- Concurrent access attempts to such resources must be *serialised*.



Two nodes, i and i + 1, being removed simultaneously result in node i + 1 not being removed.

Replica management

- Process replication supports fault tolerance and improve system availability.
- Data replication used to increase availability, balance load for better performance and hide communication latency.
- However, replication requires that the replicas be appropriately updated.
- Update cannot be done instantaneously, therefore inconsistent replicas may occur.

Failures

- A system that fails is not adequately providing the services it was designed for.
- Several classification schemes have been developed to determine how serious a given failure actually is.

Type of failure	Description of server's behavior
Crash failure	Halts, but is working correctly until it halts
Omission failure Receive omission Send omission	Fails to respond to incoming requests Fails to receive incoming messages Fails to send messages
Timing failure	Response lies outside a specified time interval
Response failure Value failure State-transition failure	Response is incorrect The value of the response is wrong Deviates from the correct flow of control
Byzantine failure	May produce arbitrary responses at arbitrary times. Unlike other failures, it is difficult to detect.

Different types of failures.

Synchronisation

- Each process in a distributed system has a local clock.
- If these clocks are all correct, the system can exhibit synchronous behaviour.
- Unfortunately, this is difficult to achieve because drift in local clocks is inevitable.
- One solution is to use a time server that keeps all local clocks synchronised (resynchronisation).

Example

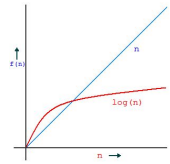
- In Unix systems the *make* command is used to compile new or modified code without the need to recompile unchanged code.
- The make command uses the clock of the machine it runs on to determine which source files need to be recompiled.
- If the sources reside on a separate file server and the two machines have unsynchronised clocks, the make program may not produce the correct results.

Global state

- The global state of a distributed system consists of the local states of its component processes.
- To compute the global state at a given time one has to read the local states of every component process at that time.
- Given the facts that local clocks are never perfectly synchronised and message propagation delays exist, computation of the global state is a nontrivial problem.

Scalability

- System is **size scalable** if can add more users/processes without impairing performance.
- Additional resources required to cope with the increased size should be manageable.
  - Size scalability **excellent** if space/time complexity is O(log n) or lower where n is the number of system processes.
  - Size scalability **poor** if space/time complexity is O(n) or higher where n is the number of processes.



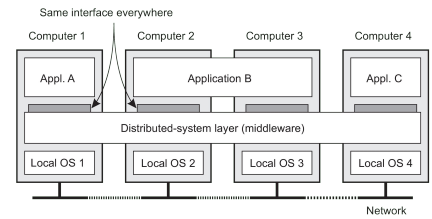
- System is **geographical scalable** if users/resources lie far apart but communication delays hardly noticed.

Pitfalls

- Implementing a distributed system is a formidable task.
- False assumptions commonly made when developing a distributed system:
  - The network is reliable.
  - The network is secure.
  - The network is homogeneous.
  - The topology does not change.
  - Latency is zero.
  - Bandwidth is infinite.
  - Transport cost is zero.
  - There is one administrator.

Middleware

- Distributed system often have separate layer of software logically placed on top of operating systems.
- Middleware is to a distributed system what an operating system is to a computer.



Distribution transparency

- Achieving **distribution transparency** is an important goal of a distributed system.
- Aim is to make the distribution of processes and resources transparent, that is invisible, to users and applications.

Type of transparency	Description
Access transparency	Hide differences in data representation and how an object is accessed.
Location transparency	Hide where an object is located.
Migration transparency	Hide that an object may be moved to another location.
Replication transparency	Hide that an object is replicated.
Concurrency transparency	Hide that an object may be shared by several independent users.
Failure transparency	Hide the failure and recovery of an object.

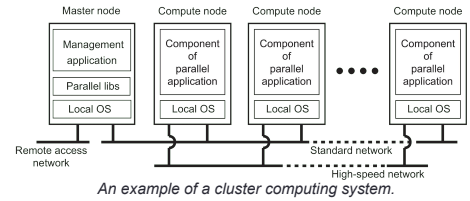
Different forms of transparency in a distributed system. An object can be a resource or a process.

Distributed vs. Parallel systems

- Any system in which the events can at best be partially ordered is a parallel system.
  - Consider three events a, b and c.
  - A total order of these events is a<b<c (< indicates must happen before).
  - A partial order of these events is a<c.
- According to this view, distributed systems form a subclass/subset of parallel systems where the state spaces of processes do not overlap.

Cluster computing

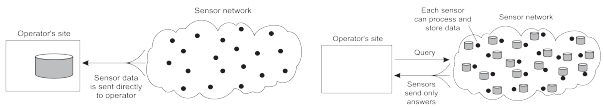
- Collection of compute nodes (computers) controlled and accessed by means of a single master node.
- Nodes have similar hardware and OS.
- Connected by high-speed local-area network and each node performs the same task.



An example of a cluster computing system.

Sensor networks

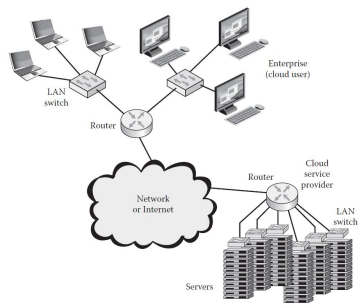
- Consists of large number of relatively small nodes, each equipped with sensors and often actuators.
- Many sensor networks use wireless communication, and the nodes are often battery powered.



Organising a sensor network database, while storing and processing data (a) only at the operator's site or (b) only at the sensors.

Cloud computing

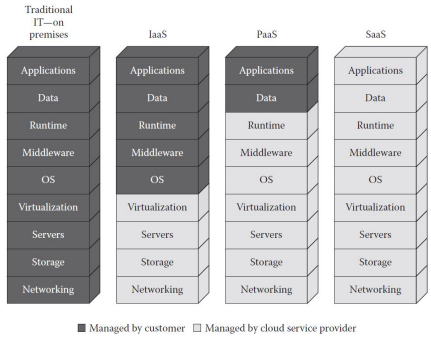
- Clients outsource their software usage, data storage and computing infrastructure to third-party companies.
- Clients need not acquire or maintain expensive hardware, software or technical staff.
- Pay-per-use basis; *measured service* model.
- Main advantages of convenience, cost reduction and elasticity.
- Presents unique security challenges. Richard Stallman referred to cloud computing as "careless computing".



Cloud service context (taken from Vacca 2016)

There are three main types of cloud computing services:

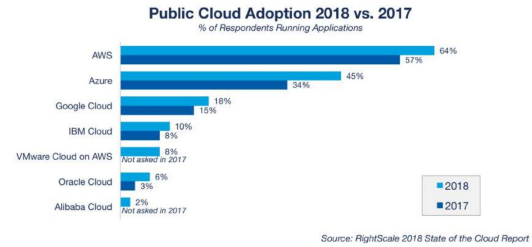
- Software-as-a-service (SaaS) – allows the usage of cloud apps e.g. Google Docs.
- Platform-as-a-service (PaaS) – virtualized environment with OS installed is rented.
- Infrastructure-as-a-service (IaaS) – includes hardware, storage, IP addresses and firewalls.



Separation of responsibilities in cloud operation (taken from Vacca 2016)

Cloud migration

- Cloud migration is the process of moving to a cloud computing environment.



Source: RightScale 2018 State of the Cloud Report

- There are different approaches to cloud migration; each with corresponding pros and cons.

	Lift and shift (Rehost) *The application moves from on premises to cloud "as is"*	**Rearchitect (Refactor)** *The application undergoes architectural and/or code changes before it moves to cloud*
PROS	▪ Requires little upfront effort in migration process ▪ Fast to migrate and deploy	▪ App takes full advantage of cloud-native features and benefits ▪ App cost-effectively runs in cloud
CONS	▪ App is unable to take full advantage of cloud-native features and benefits ▪ App can cost more to run in cloud	▪ Incurs more upfront costs in migration process, and is often time-consuming and resource-intensive

Figure taken from https://whatis.techtarget.com/definition/lift-and-shift