COMP23412 Distributed Systems (50)

Distributed System is a collection of autonomous *computing elements* that appears to its users as a *single coherent system*. In a distributed system, computation is *concurrent* (more than one processes), and there is no *shared state* (global clock). It is important to remember that not all nodes are created equal, they may have different costs, different capabilities, and all sorts of things.

Fallacies in Distributed Systems:

1. The Network is Stable Connectivity between the nodes can vary, as they are developed by different organisations using different hardware and software. The final result will be a network of networks.

2. The Network is Secure Security must be built into the system since day 1.

3. The Network is *Homogeneous* All nodes may have different specs and capabilities.

4. Topology does not Change. Many things would cause the network topology to change.

5. Latency is Zero. The minimum round-trip time between two points on earth is determined by speed of light, further the node is, higher the latency.

6. Bandwidth is Infinite. There is a limit on *throughput* (data successfully transferred in a given timeframe)

7. Transport Cost is Zero. It costs a lot of parties to transport data across the Internet.

8. There is One Administrator. There are many, usually one per node.

Distributed System Layer (Middleware) is designed to facilitate inter-application communication, security, transaction management, and recovery from failures, and many more. It exists between the applications and the local, individual OSes.

Two-Phase Commit works by sending all the commands to all the nodes, and then commit the changes if everything is good to go. It could prevent many failures, but may still require manual actions to recover from failure.

Mutual Exclusion is to prevent race conditions. A program would not enter a critical section if another program is already in one. It also allows two programs to access the same resources concurrently.

Lamport's Logical Clock assumes the message would be delivered in one clock tick, and that an event must happen before it is received. This is the add 1 method.

Cristian's Algorithm would change the time of the requester's time to remote time T+(RTT/2). This assumption may not always hold.

Berkeley Algorithm sync time on multiple servers by averaging the not-so-outrageous times using Cristians's algorithm and send back to all servers the offset they need to adjust.

Naming Approaches for nodes across a network – ***Centralised***, where a single point of failure deal with every request; ***Free-for-All***, where nodes across a network decide its own address; ***Delegation Naming Responsibilities***, where an authority allocate names to smaller systems.

SMTP is a protocol where multiple communications is needed to complete one transaction. It is an old and outdated email protocol.

Memory and Storage is managed by the operating system, and the allocation may not be entirely managed by the user or process initiated the program that is requesting it.

Sequential Processing is a way of processing data that ensures that all operations are performed in a specific order. It helps to ensure that data is consistent and that there are no conflicts between different processes.

Concurrency in program execution needs support from program as well as hardware.

Structured P2P Systems where nodes are organized in an overlay that adheres to a specific, deterministic topology: a ring, a binary tree, a grid, etc.

Unstructured P2P System where nodes maintain an ad-hoc list of neighbours, resulting an overlay resembles a random graph. A node generally needs to constantly update its local list, and need to search for all data it needs.

Low-Level Operations, send and receive, do not conceal communications.

Remote Procedure Call (RPC) could hide most of the complexities, and is ideal for client-server applications. Marshalling is a must before inputs could be sent through RPC, which is a complex procedure.

Message-Oriented Middleware (MOM) is a message queue that caches all the messages it receives, and wait for the processing unit to pick them up.

Data Replication is used to enhance system ***reliability*** *(always online)*, and to improve the ***performance*** *(response time from system)*. It can also potentially improve system's ***scalability*** *(ability to increase capability)*. But it will pose challenge to keep replicas consistent. However, it potentially can damage user's trust in quality of data (not always the most recent), and could slow things down (to obtain copy).

Sequential Consistency guarantees that all processes/machines observe all operations in the same order.

Casual Consistency only make sure operations potentially casually related to be in the same order. However, concurrent writes may be seen in a different order on different machines/processes.

Security in Distributed Systems can be divided into: 1) having a secure communication channel, and 2) having proper access control mechanisms.

Confidentiality ensures information is disclosed only to authorized parties.
Integrity ensures alterations to a system's assets can be made only in an authorised way.

Auditing Tools are used to trace which clients accessed what, and in which way.
Encryption can provide an appropriate safeguard against the unauthorised processing of personal data.

Fault Tolerance is enabling the system to remain operational in an acceptable way, despite failures.

Availability: the probability that the system operates correctly at any given moment.
Reliability: length of time that it can run continuously without failure.
Safety: if and when failures occur, the consequences are not catastrophic for the system.
Maintainability: how easily a failed system can be repaired.

Types of Redundancy:

- ***Physical Redundancy*** where spare servers available and could be switched to when needed.
- ***Time Redundancy*** where an action is granted time to perform an action multiple times.
- ***Information Redundancy*** sends extra bits or information during transmission to allow recovery.

ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*) of a database transaction:
***Atomicity*:** Either all occur, or nothing occurs.
***Consistency*:** Leave the database in a consistent state.
***Isolation*:** Does not affect the execution of other transactions.
***Durability*:** Effects must be permanent, even in the event of a system failure.

Byzantine Failures are failures that occur at arbitrary times. It would be the most serious type of failures.