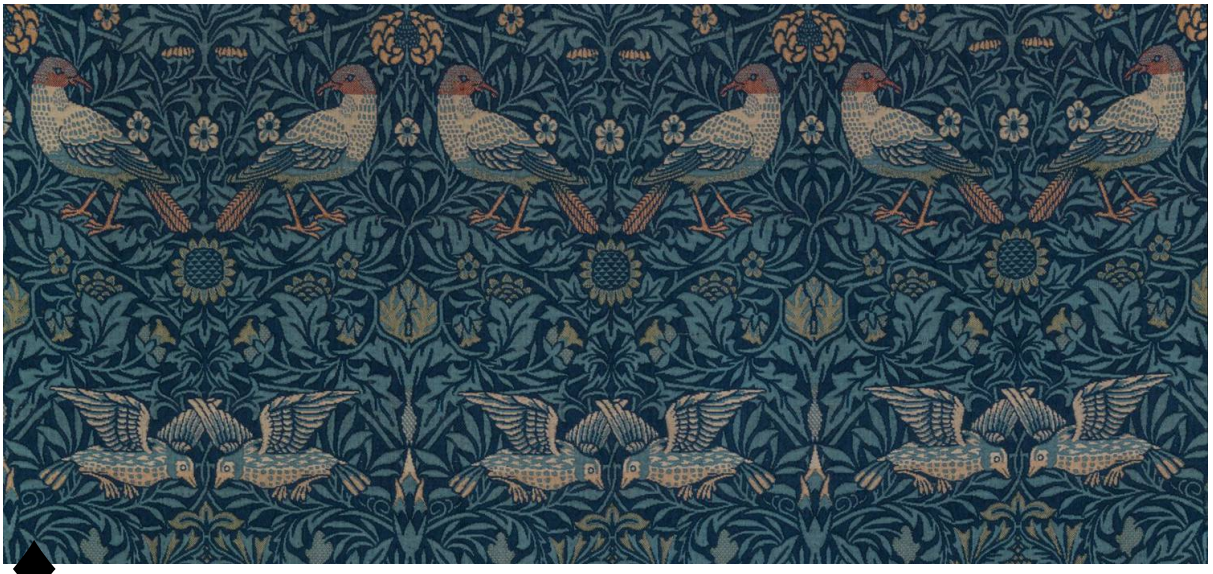


Distributed Systems

Through *Maya Mnaizel*

Distributed Systems are complex

Perspectives on Distributed Systems:

1. Architecture
2. Process
3. Communication
4. Coordination
5. Naming
6. Consistency
7. Fault Tolerance
8. Security

What do we want to achieve in Distributed Systems?

- Support sharing of resources
- Distribution transparency

- Openness
- Scalability

Distribution Transparency

Transparency:

The phenomenon wherein a distributed system makes an effort to conceal the reality that its resources and activities are spread physically over a number of computers, perhaps separated by great distances.

Distribution transparency is managed at a layer between operating systems and applications using a variety of techniques, including: a middleware layer

7 Layers of Distribution Transparency

- Access
  - Cover up variations in data representation and object access methods
- Location
  - Hide where an object is located
- Relocation
  - Conceal the possibility that an object might be moved while being used
- Migration
  - Conceal the possibility that an object might be moved to another location
- Replication
  - Hide that an object is replicated
- Concurrency
  - Conceal that multiple separate users may share an objects

- Failure
  - Hide the failure and recovery of an object

It is impossible to hide failures completely of networks and nodes

Full transparency will cost performance

Conclusion

Distribution Transparency is nice to have, but achieving it is a different story

---

Openness of Distributed Systems

A system that provides parts that other systems can quickly use or integrate. The components that make up an open distributed system itself are frequently imported from outside

So systems should be:

- Conform to well-defined interfaces
- Effortlessly collaborate
- support portability
- easily extensible

Dependability

Basics

Tightly coupled: a component may depend on some other component

Specifically

Loosely coupled: components are processes or channels

Requirements related to dependability

- Availability
- Reliability
- Safety
- Maintainability

Terminology

Term	Description	Example
Failure	A component is not living up to its specifications	Crashed program
Error	Part of a component that can lead to a failure	Programming bug
Fault	Cause of an error	Sloppy programmer

Handling faults

Term	Description
Fault Prevention	Prevent the occurrence of a fault
Fault Tolerance	Built a component and make it mask the occurrence of a fault

Fault Removal	Reduce the presence or seriousness of a fault
Fault Forecasting	Estimate current presence, future incidence and consequences of faults

---

Security

We need secure distributed systems so it can be dependable

What we need:

- ▼ Confidentiality

  Information is disclosed only to authorized parties

- ▼ Integrity

  Prevent deletion or modification in an unauthorized way

- ▼ Authentication

  Verifying the claimed identity

- ▼ Authorization

  Identifying if the entity has proper access rights

- ▼ Trust

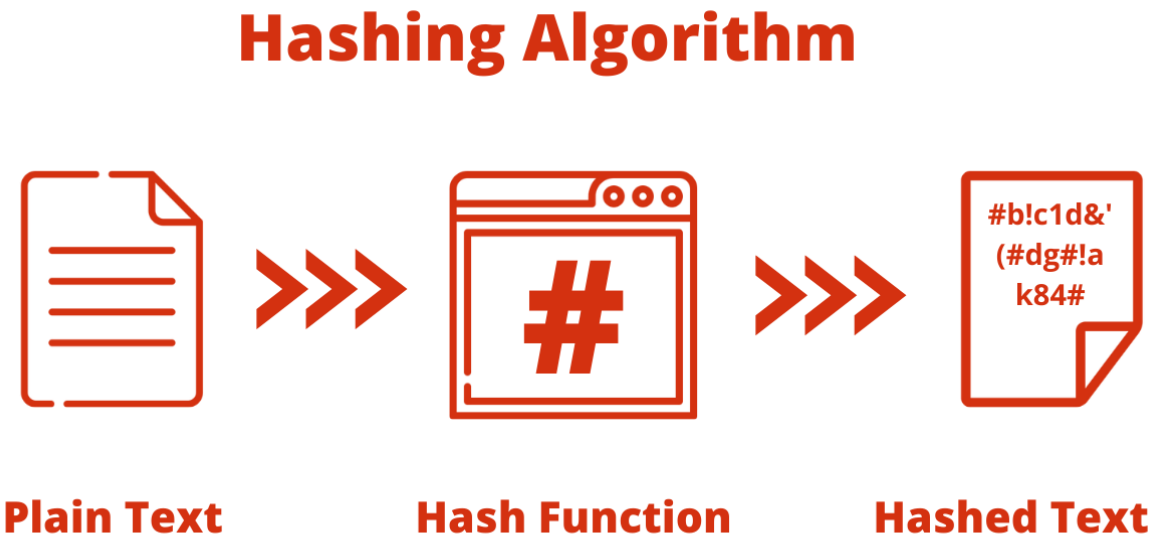
  perform actions according to a specific expectation

Security Mechanisms

- Symmetric Cryptosystem
- Asymmetric Cryptosystem

https://th.bing.com/th/id/R.4f621ab51b8445ba162ce057a1543cd2?rik=RmG9aqrTDHyWXA&riu=http%3a%2f%2ftechdifferences.com%2fwp-content%2fuploads%2f2016%2f09%2fSymmetric-Vs-Asymmetric-Encryption.jpg&ehk=j5XaaURHMK5CHKYyTQYQnNBDN%2bR83q8tm%2fuTqjY%2bplQ%3d&risl=&pid=ImgRaw&r=0

- Security Hashing



Scale in Distributed Systems

Numbers of users or processes (Size Scalability)

Maximum distance between nodes (Geographical Scalability)

Number of administrative domains (Administrative Scalability)

Root causes for scalability problems with centralized solutions

- The computational capacity is limited by the CPUs

- The storage capacity, including the transfer rate between CPUs & disks
- The network between the user and the centralized service

Problems with Geographical Scalability

- Cannot simply go from LAN to WAN
- WAN links are often inherently unreliable
- Lack of multipoint communication

Problems with Administrative Scalability

- Conflicting policies concerning usage, management, and security, e.g.,
  - Computational Grids: shared expensive resources between different domains
  - Shared Equipment: how to control, manage, and use a resource

Exception:

Several peer-to-peer networks

- File-sharing systems
- Peer-to-peer telephony
- Peer-assisted audio streaming

Techniques for Scaling

Hide Communication Latencies

- make use of asynchronous communication
- have separate handler for incoming response

Partition Data and Computations across Multiple Machines

- move computations to clients
- Decentralized naming service (DNS)
- Decentralized information systems (WWW)

Replication and Caching

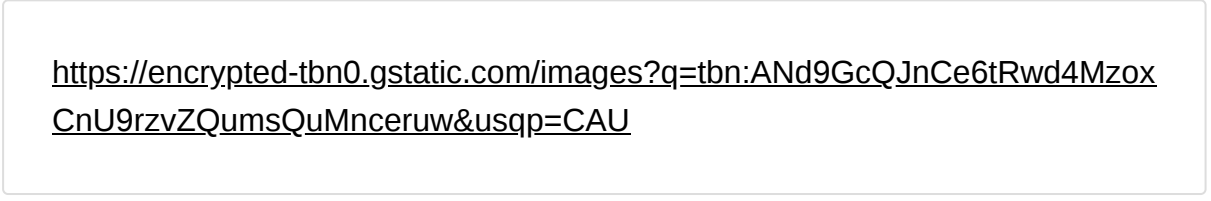
- Replicated files servers and databases
- Mirrored websites
- Web caches
- File caching

> 💡 Having multiple copies leads to inconsistencies modifying one copy makes that copy different from the rest, requiring global synchronization on each modification

---

Computing Architecture

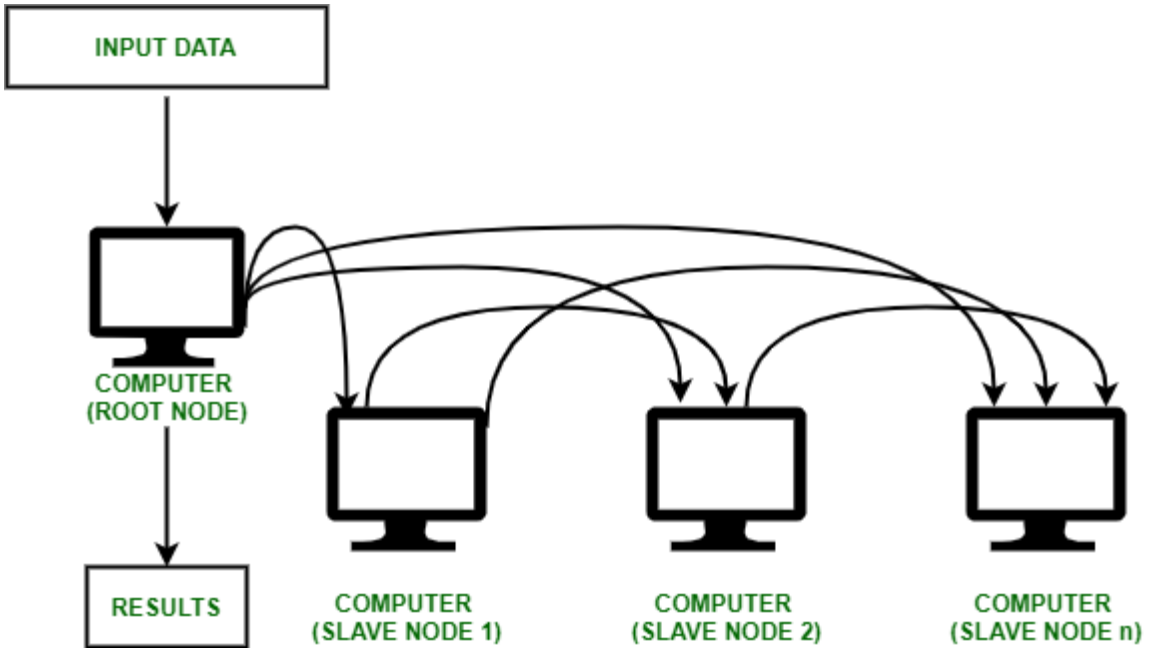
Parallel Computing


https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQJnCe6tRwd4MzoxCnU9rzvZQumsQuMnceruw&usqp=CAU

- Compared to multicomputers, multiprocessors are easier to program, but as the number of processors (or cores) increases, issues arise.

Resolutions: attempt to use a shared-memory model on top of multicomputer

Cluster Computing



A collection of upscale computers linked by LAN

Homogeneous: one or more tightly coupled managing nodes; nearly identical hardware and the same operating system

Grid Computing

https://www.researchgate.net/publication/298215186/figure/fig5/AS:669391670759436@1536606819748/A-layered-architecture-for-grid-computing-systems.ppm

Loosely coupled

- Heterogeneous
- Distributed throughout multiple organizations
- Easily able to traverse a vast network

EAI

Allow direct application-to-application communication, leading to Enterprise Application Integration (EAI)

Transaction

BEGIN_TRANSACTION	Start transaction
END_TRANSACTION	Terminate transaction
ABORT_TRANSACTION	Kill transaction
READ	Read data
WRITE	Write data

Issues : all-or-nothing

Atomic	happens seemingly
Consistent	does not violate system invariants
Isolated	not mutual interference
Durable	commit means changes are permanent

TPM

https://www.researchgate.net/publication/298215186/figure/fig7/AS:669391670738969@1536606819895/The-role-of-a-TP-monitor-in-distributed-systems.ppm

---

Middleware as a communication facilitator in enterprise application integration.

https://www.researchgate.net/profile/Maarten-Van-Steen/publication/298215186/figure/fig8/AS:669391674957824@1536606820042/Middleware-as-a-communication-facilitator-in-enterprise-application-integration.ppm

Remote Procedure Call (RPC)

Requests are made via a local procedure call, which is then processed, packaged as a message, answered via a message, and the outcome is returned as a call return

Message-Oriented Middleware (MOM)

The logic contact point (published) receives messages, which are then forwarded to applications that have subscribed

---

Distributed Pervasive Systems

Three Subtypes

- Ubiquitous Computing Systems
- Mobile Computing Systems
- Sensors and networks

Ubiquitous Systems

- Devices are networked, distributed, and accessible transparently
- Interaction between users and devices is highly invisible
- The system is aware of a user’s context to optimize interaction
- Devices operate autonomously without human intervention
- The system as a whole can handle a wide range of dynamic actions and interactions

Mobile Systems

Distributed systems involve numerous mobile devices whose locations change over time, affecting local services and reachability. Maintaining stable communication can be problematic. While past research focused on direct resource sharing between mobile devices, this approach is now considered unfruitful.

Sensor Networks

Characterized by numerous simple nodes (ranging from tens to thousands) with limited memory, compute, and communication capacity, often powered by batteries or even without batteries.

Reference

S. Tanenbaum, Andrew, and Maarten van Steen. “Distributed Systems 4th Edition (2023).” DISTRIBUTED-SYSTEMS.NET, 1 Aug. 2023, www.distributed-systems.net/index.php/books/ds4/.

Questions

1. **What does distribution transparency hide?**
    a. Location
    b. Relocation
    c. Migration
    d. Replication
    e. Concurrency
    f. Failure
2. **Is full distribution transparency achievable?**

No, it may be too much and there are limitations

3. **What are some limitations of achieving full distribution transparency?**
    a. Communication latency
    b. Inability to completely hide failures
    c. Inability to distinguish a slow computer from a failing one
    d. Uncertainty or server operations before crash
    e. Performance cost
    f. Time to keep replicas up-to-date
    g. The need to immediately flush write operations to disk for fault tolerance
4. **What are some common misconceptions about centralized solutions?**

they do not scale

5. **What is the root of the Domain Name System?**

Logically Centralized and physically distributed (massively Distributed)

6. **Do centralized solutions always have a single point of failure?**

No, they often do not have a single point of failure. They can be easier to manage and more robust

7. **What should we develop skills in to judge misconceptions about distributed systems?**

We need to develop skills in understanding distributed systems

8. **What are the perspectives to consider when studying distributed systems?**
    a. Architecture
    b. Process
    c. Communication
    d. Coordination
    e. Naming
    f. Consistency
    g. Replication
    h. Fault tolerance
    i. Security
9. **What are the overall design goals of distributed systems?**
    a. Support sharing of resources
    b. Distribution transparency
    c. Openness
    d. Scalability
10. **What is the degree of transparency when aiming at full distribution transparency?**

Full transparency is not achievable due to communication latencies and the impossibility of completely hiding network and node failures.

11. **Why is full transparency not achievable in distributed systems?**

You cannot distinguish a slow computer from a failing one, and you can never be sure that a server actually performed an operation before crashing.

12. **What is the cost of achieving full transparency in a distributed system?**

Will cost performance and expose the distribution of the system

13. **When may exposure to distribution be beneficial in a distributed system?**
    a. When making use of location-based services
    b. Dealing with users in different time zones

c. Making it easier for a user to understand what's going on

14. **What is an open-distributed system?**

It offers components that can easily be used by or integrated into other systems, It consists of components from various sources

15. **What are the goals of implementing openness in distributed systems?**

a. Conforming to well-defined interfaces

b. Supporting interoperability

c. Enabling portability of applications

d. being easily extensible

16. **What are the policies involved in implementing openness?**

a. Determining the level of consistency for client-cached data

b. The operations allowed for downloaded code

c. The adjustments of QoS requirements in varying bandwidth

d. The level of secrecy for communication

17. **What are the mechanisms involved in implementing openness?**

a. Allowing dynamic setting of caching policies

b. Supporting different level of trust for mobile code

c. Providing adjustable QoS parameters per data stream

d. Offering different encryption algorithms

18. **What is the average time until a component fails?**

Mean Time To Failure (MTTF)

19. **What is the average time needed to repair a component?**

Mean Time To Repair (MTTR)

20. **What is the average time between failures?**

Mean Time Between Failures (MTBF)

21. **What is a failure?**

A component is not living up to its specifications

22. What is an error?

Part of a component that can lead to a failure

23. **What is a fault?**

Cause of an error

24. **What is fault prevention?**

prevent the occurrence of a fault

25. **What is fault tolerance?**

Build a component and make it mask the occurrence of a fault

26. **What is fault removal?**

Reduce the presence, number, or seriousness of a fault

27. **What is fault forecasting?**

Estimate current presence, future incidence, and consequences of faults

28. **What is confidentiality?**

Information is disclosed only to authorized parties

29. **What is integrity?**

Ensure that alterations to assets of a system can be made only in an authorized way

30. **What is authentication?**

Verifying the correctness of a claimed identity

31. **What is authorization?**

Does an identified entity have proper access rights

32. **What is trust?**

One entity can be assured that another will perform particular actions according to a specific expectation

33. **What level of consistency do we require for client-cached data?**

Consistency level

34. **What is the difference between distributed and decentralized systems?**

Distributed systems have no central authority, while decentralized systems have multiple authorities

35. **When does a decentralized system become distributed?**

when links are added between nodes in a decentralized systems

36. **What are the two views on realizing distributed systems?**
    a. Integrative: involves connecting existing networked computer systems into larger systems
    b. Expansive: extends an existing networked computer system with additional computers
37. **What is an example of a system that is logically centralized but physically distributed**

The Domain Name System is an example. It is logically centralized as it provides a unified namespace, but it is physically distributed across multiple servers and decentralized across several organizations