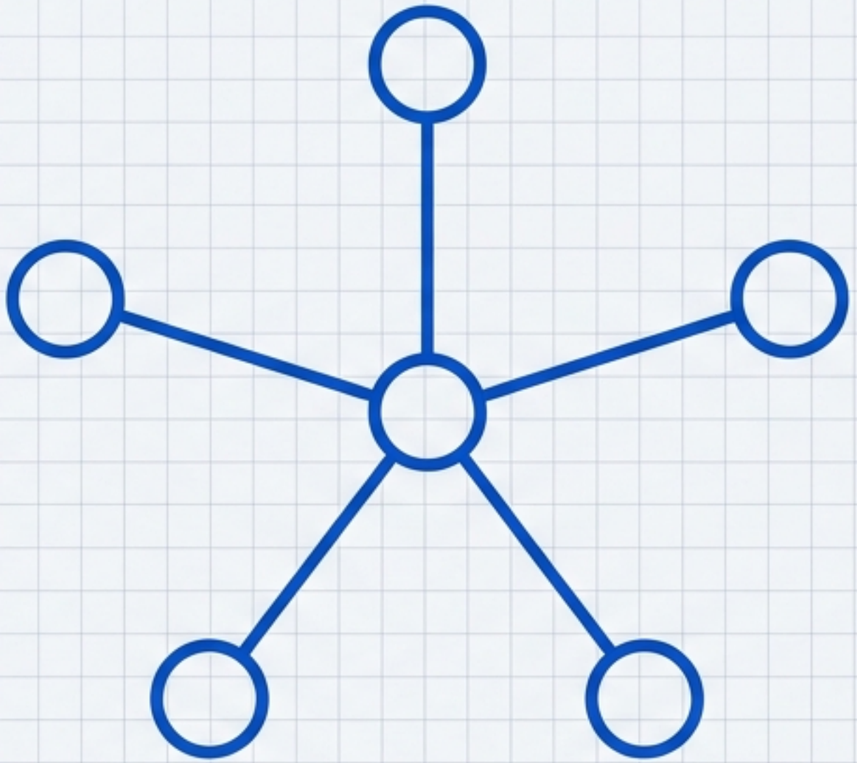
DISTRIBUTED SYSTEMS

Principles & Paradigms: The Architect’s Blueprint



Based on the text by Maarten van Steen & Andrew S. Tanenbaum

A collection of independent computers that appears to its users as a single coherent system.

Distributed Systems (Expansive View)

Concept: Spreading is Sufficient.

Driver: Expansion & Dependability (e.g., Cloud CDNs, Google Mail).

Goal: Coherence. To the user, it appears as one computer.

Decentralized Systems (Integrative View)

Concept: Spreading is Necessary.

Driver: Administrative Boundaries & Trust (e.g., Blockchain, Federated Learning).

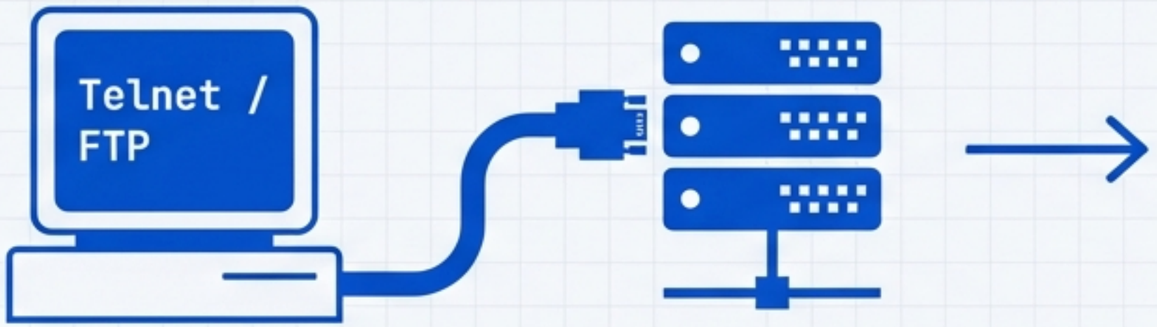
Goal: Integration. The separation is the defining feature.

Key Insight: A distributed system hides its geography; a decentralized system is defined by it.

Design Goal 1: Resource Sharing

From Explicit Connection to Seamless Integration

The Past: Remote Access



Explicit connection to foreign machines.

The Present: Seamless Integration

Cloud Storage & P2P. User places a file locally; system handles the distribution.

Key Drivers

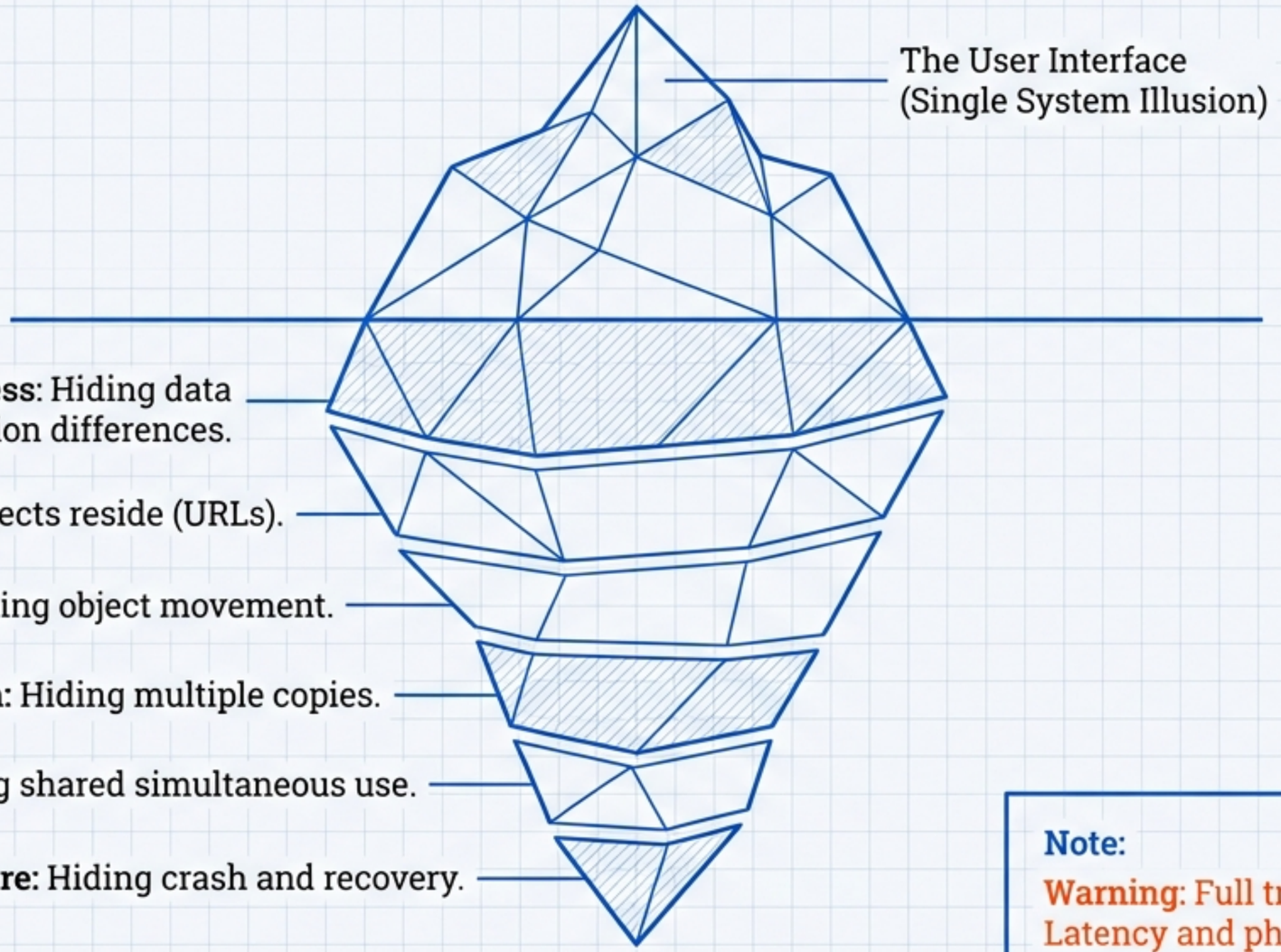
Economics

Sharing expensive peripherals (NAS) and storage infrastructure.

Collaboration

Groupware, collaborative editing, and data exchange.

Design Goal 2: Distribution Transparency



Note:

Warning: Full transparency is impossible. Latency and physics impose limits.

Design Goal 3: Openness

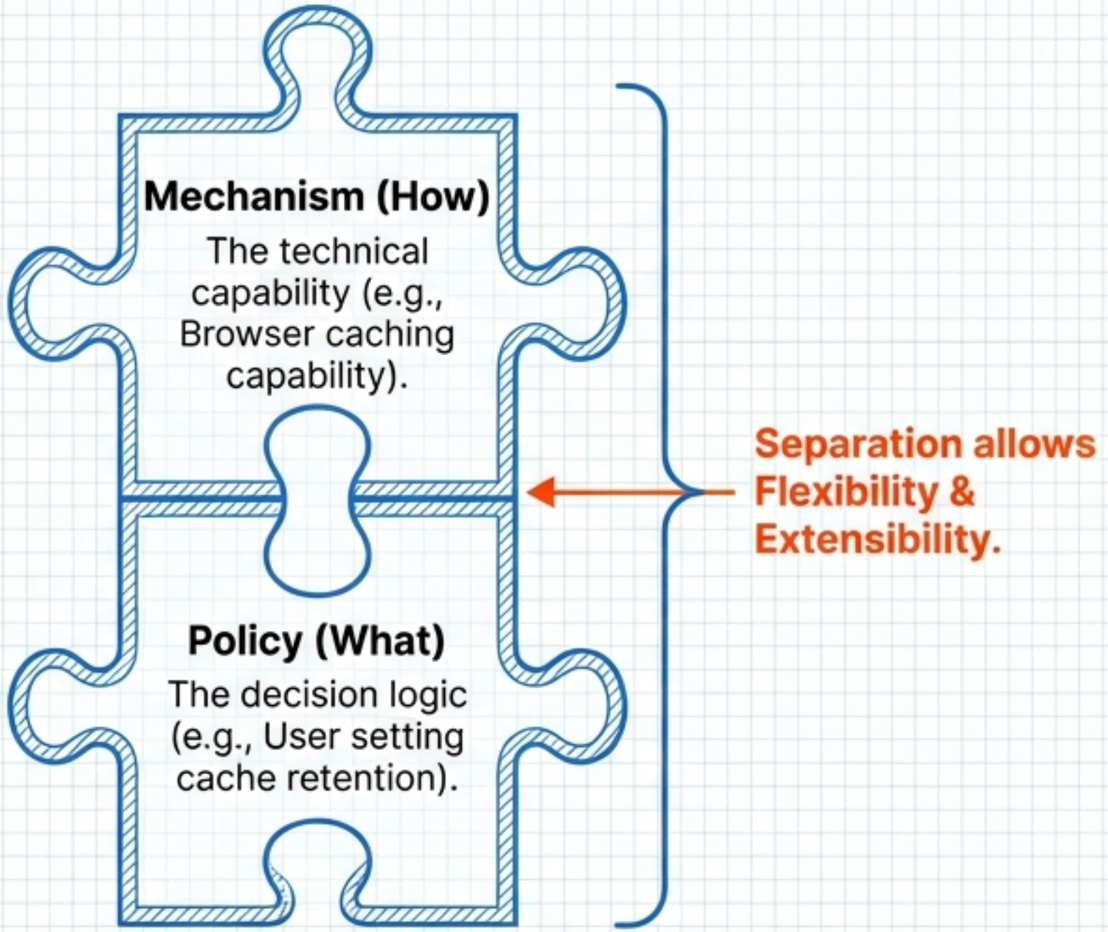
Separating Policy from Mechanism

Key Requirements

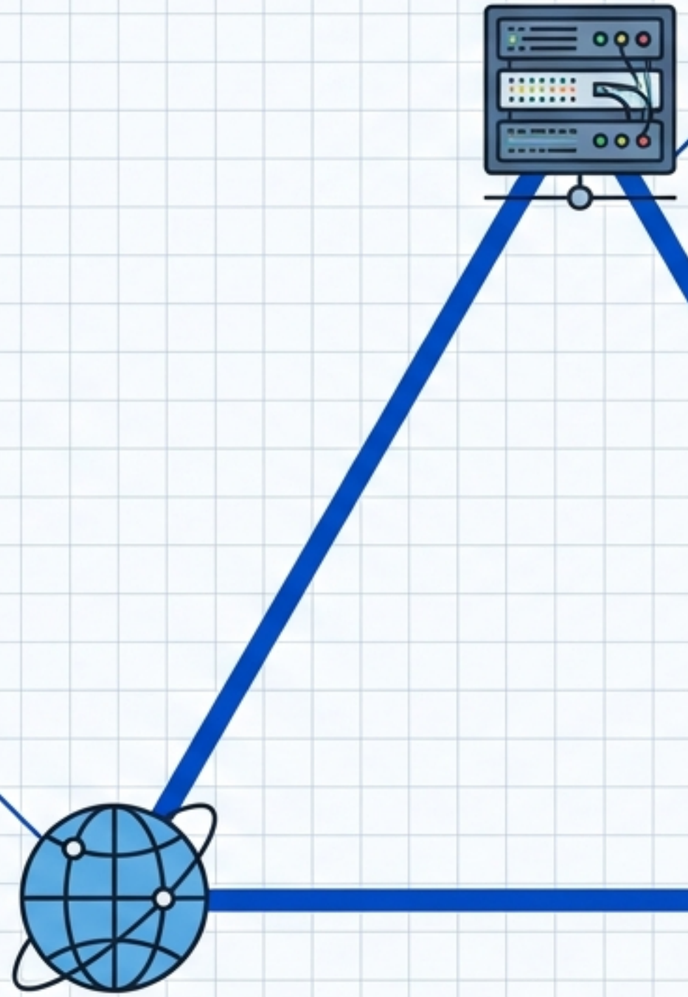
Interoperability: Implementations from different vendors co-existing.

Portability: Application mobility across systems.

IDL (Interface Definition Language): Neutral implementation syntax.



Design Goal 4: The Dimensions of Scalability



Size Scalability

Challenge: Adding users/resources.

Bottleneck: Centralized server limitations.

Geographic Scalability

Challenge: Distance between nodes.

Barrier: Latency & unreliable WAN links.

Administrative Scalability

Challenge: Crossing organizational boundaries.

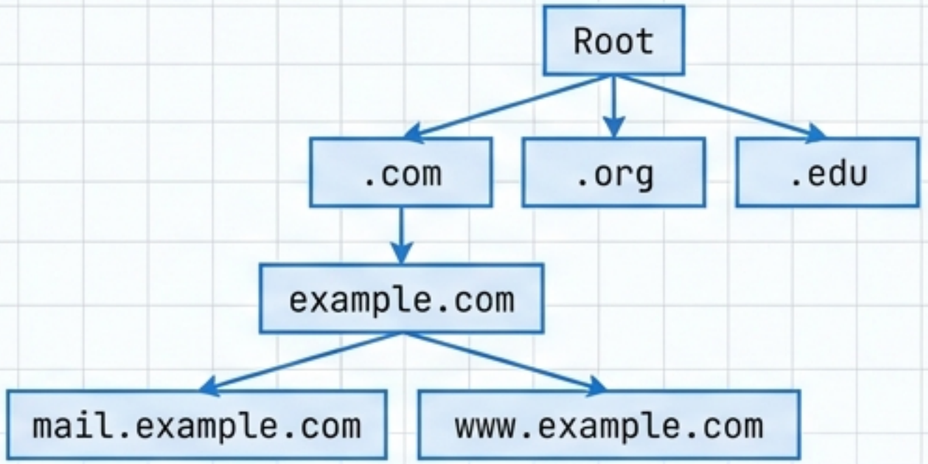
Friction: Conflicting security & payment policies.

Solving Scalability: Techniques for Expansion

Hiding Latency

- Async Communication: Fire-and-forget instead of blocking.
- Code Shipping: Move validation logic to the client (e.g., JavaScript).

Partitioning



Splitting components into smaller parts (e.g., DNS Zones, Web Sharding).

Replication

- Balancing load and increasing availability.
- **The Trade-off**: Consistency vs. Synchronization costs.

Design Goal 5: Dependability & Fault Tolerance

The Four Pillars of Dependability

Availability:
Ready for immediate use?

Reliability:
Continuous operation without failure?

Safety:
Catastrophe avoidance upon failure.

Maintainability:
Ease of repair.

The Chain of Causality

Fault (The Cause)
Bug, loose wire, cosmic ray.

↓

Error (The State)
Incorrect bit value or internal state.

↓

Failure (The Behavior)
System fails to meet promises.

Goal: Failure Masking. Providing service despite faults.

Design Goal 6: Security

The CIA Triad

Confidentiality
Authorized disclosure only.

Integrity
Authorized alterations only.

Authentication
Identity verification.

Cryptographic Mechanics

Symmetric

Single shared key for encryption/decryption.

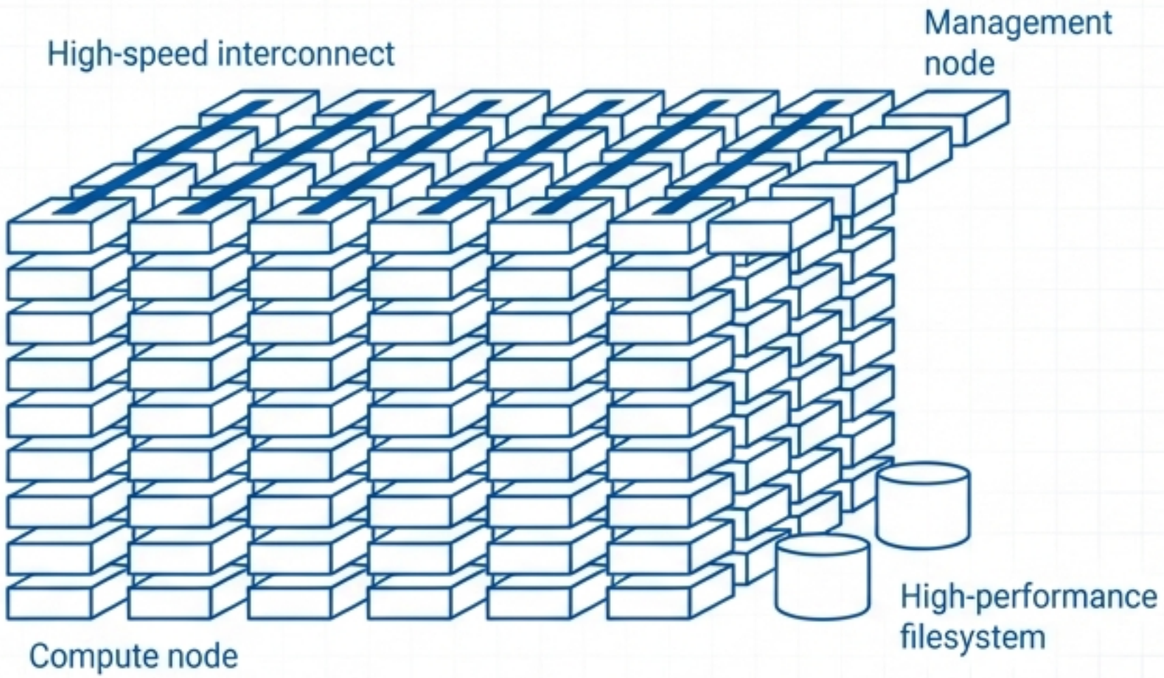
Asymmetric (Public-Key)

Public key verifies; Private key signs/decrypts

Hashing: Fixed-length strings for integrity verification.

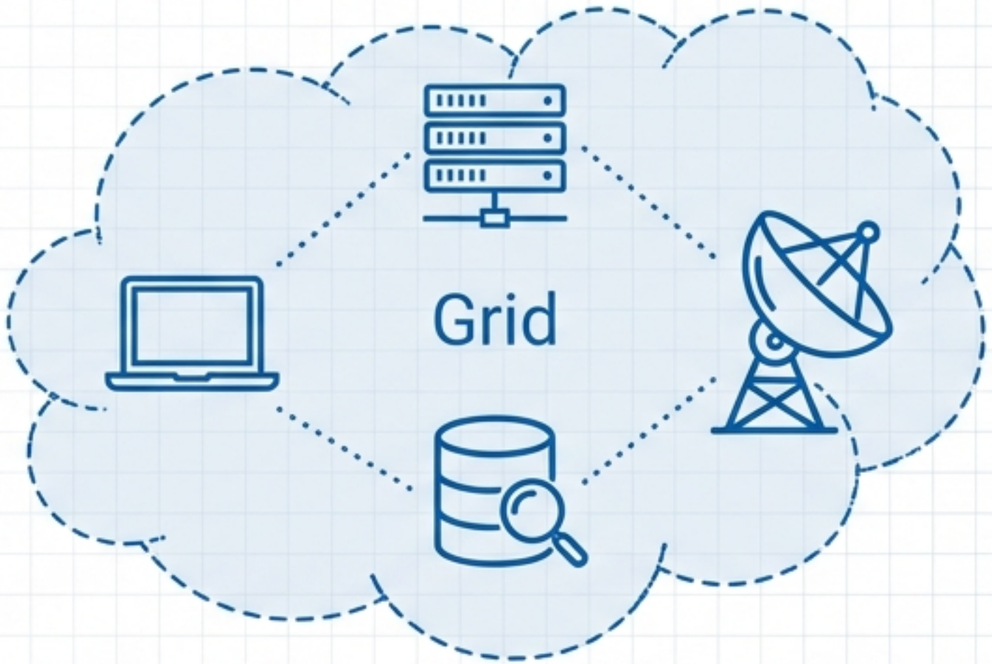
Landscape: High-Performance Computing (HPC)

Cluster Computing



- Homogeneous hardware & OS.
- Single management node.
- High-speed LAN (Tightly coupled).

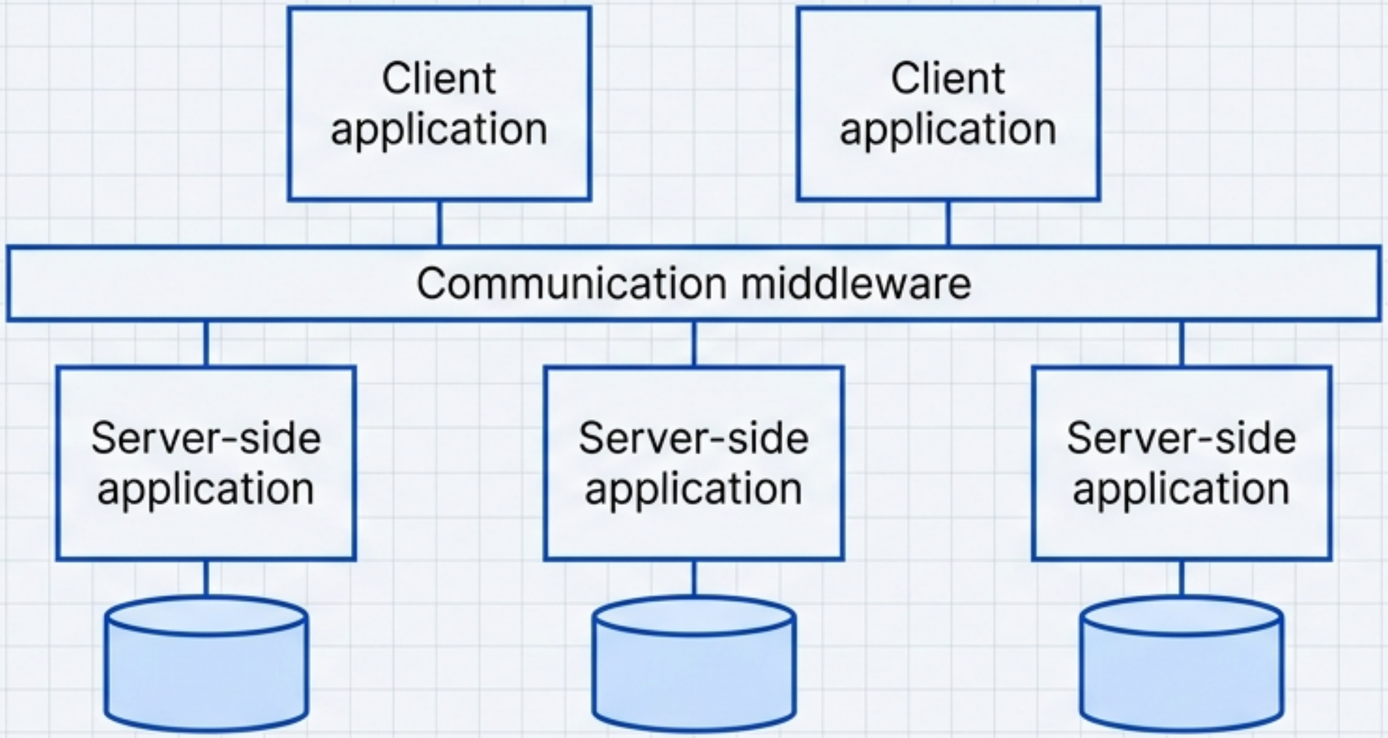
Grid Computing



- Heterogeneous hardware & OS.
- Federated: Spans administrative domains.
- Virtual Organizations.

Landscape: Distributed Information Systems

Enterprise Application Integration (EAI)



Transaction Processing (TP)

Operations adhere to ACID properties (Atomic, Consistent, Isolated, Durable).

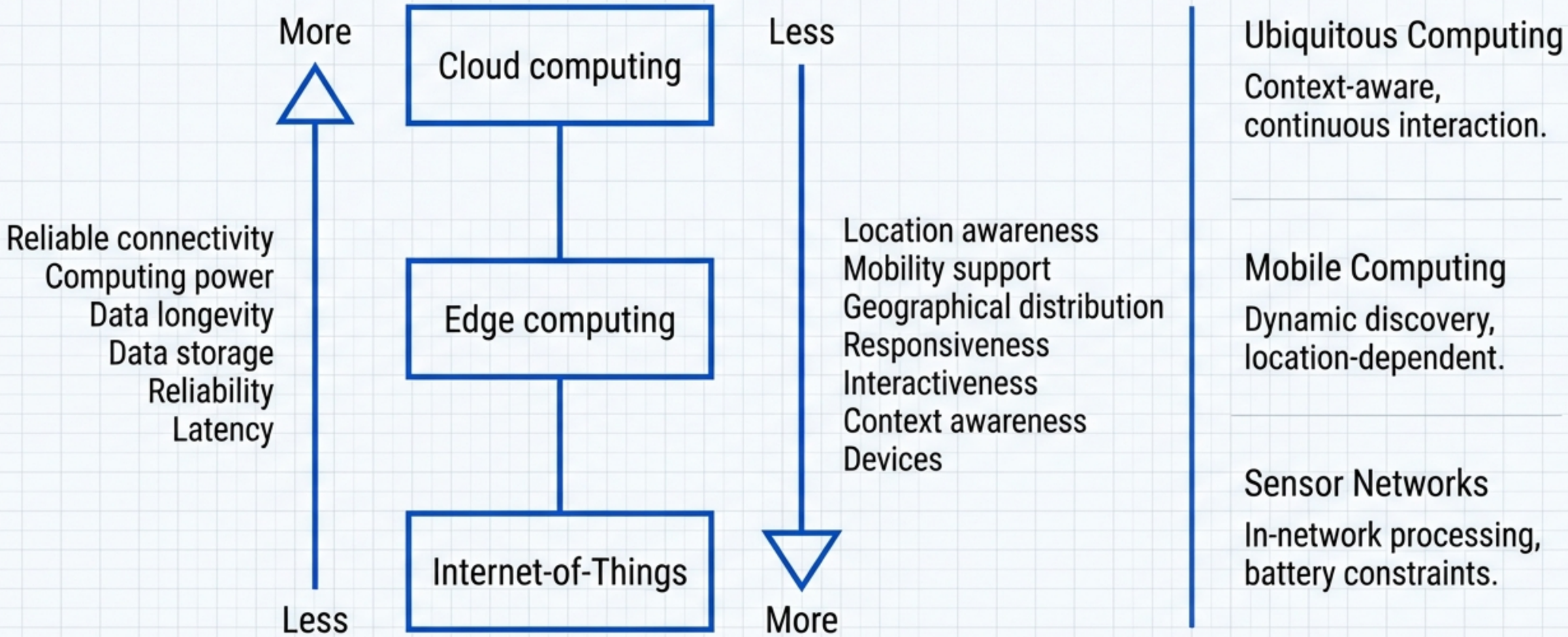
Remote Procedure Call (RPC)

Direct calls between application components.

Message-Oriented Middleware (MOM)

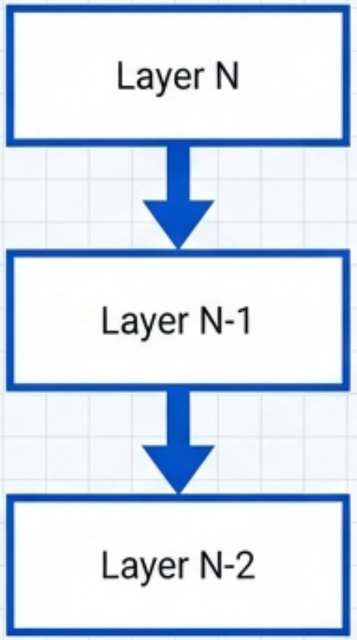
Asynchronous messaging (Publish-Subscribe).

Landscape: Pervasive Systems



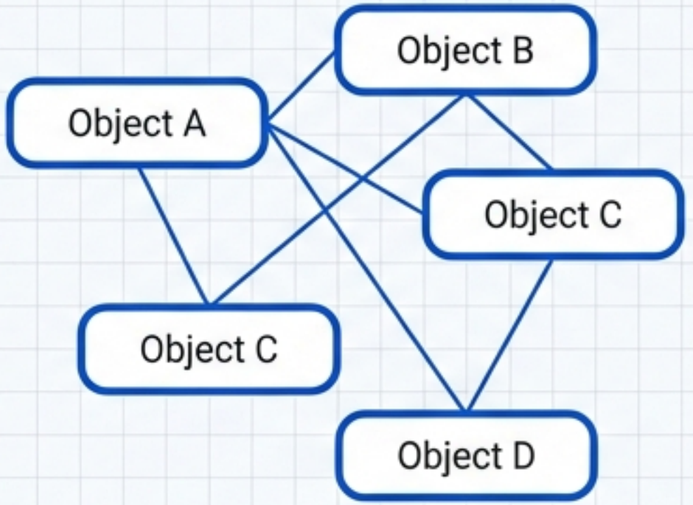
Architectural Styles: Organizing Components

Layered



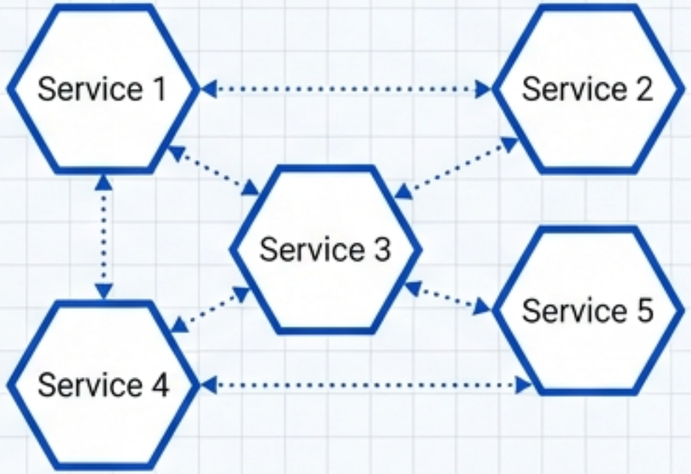
Hierarchical downcalls.
Strict separation.

Object-Based



Roboto Slab
State + Methods encapsulated.
Connected via RMI.

Service-Oriented (SOA)



Roboto Slab
Loosely coupled. "Do one thing well". Independent deployment.

The 8 Pitfalls of Distributed Design

Peter Deutsch's False Assumptions

1	✗ The network is reliable.	2	✗ The network is secure.
3	✗ The network is homogeneous.	4	✗ The topology does not change.
5	✗ Latency is zero.	6	✗ Bandwidth is infinite.
7	✗ Transport cost is zero.	8	✗ There is one administrator.

Successful design requires explicitly managing the fact that all 8 assumptions are false.