SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

Department of Computer Science and Engineering

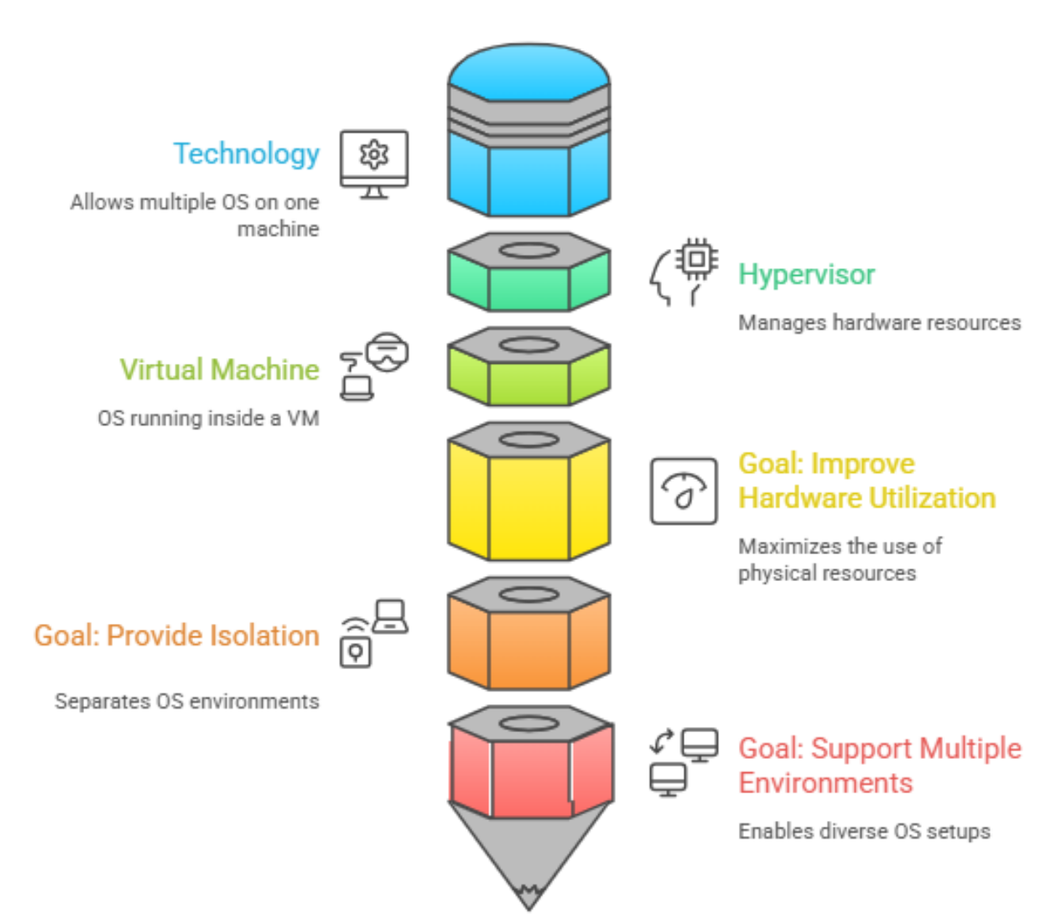
23CST206–OPERATING SYSTEMS AND VIRTUALIZATION

B.E- CSE /IV SEMESTER

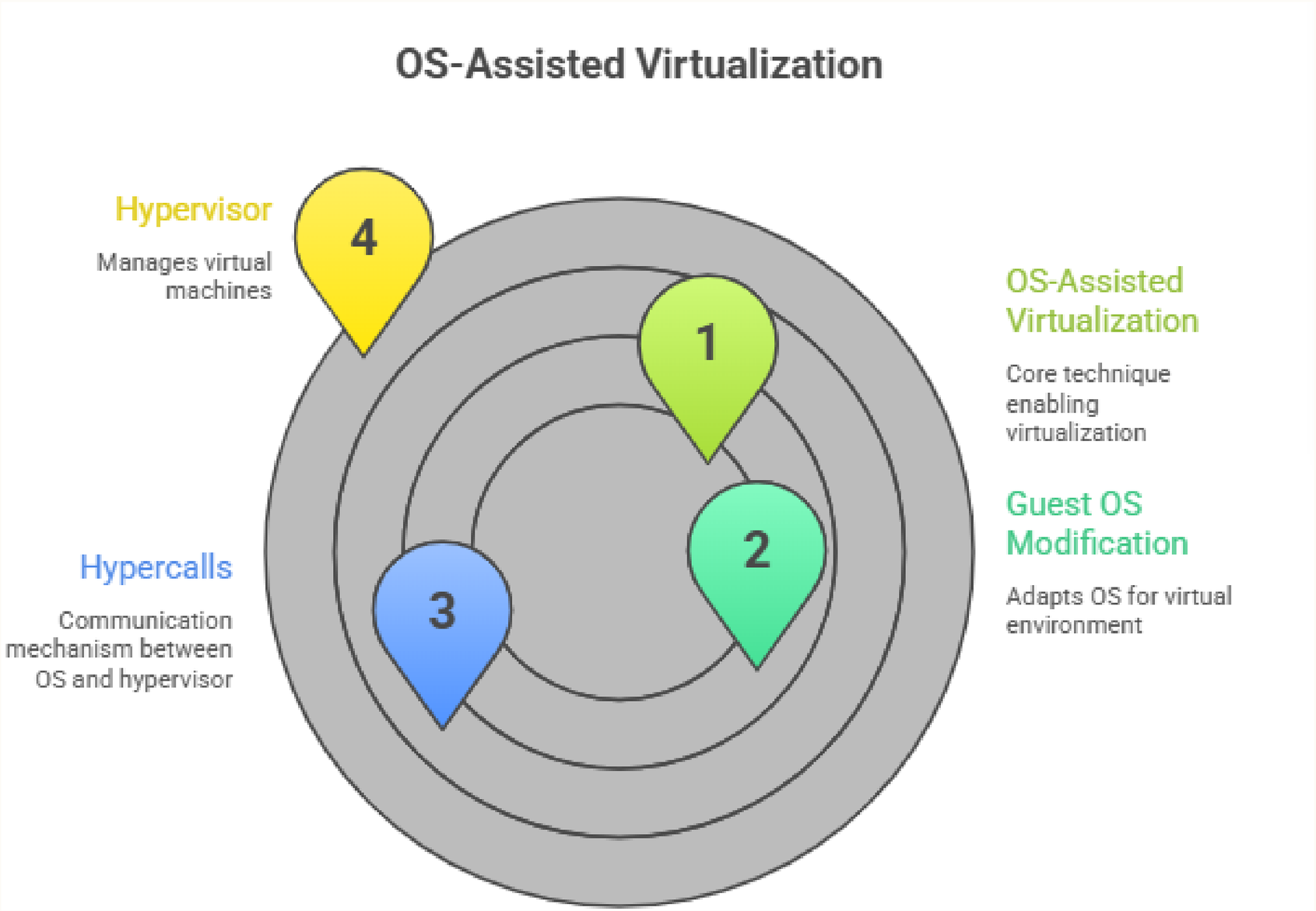
UNIT – IV VIRTUALIZATION

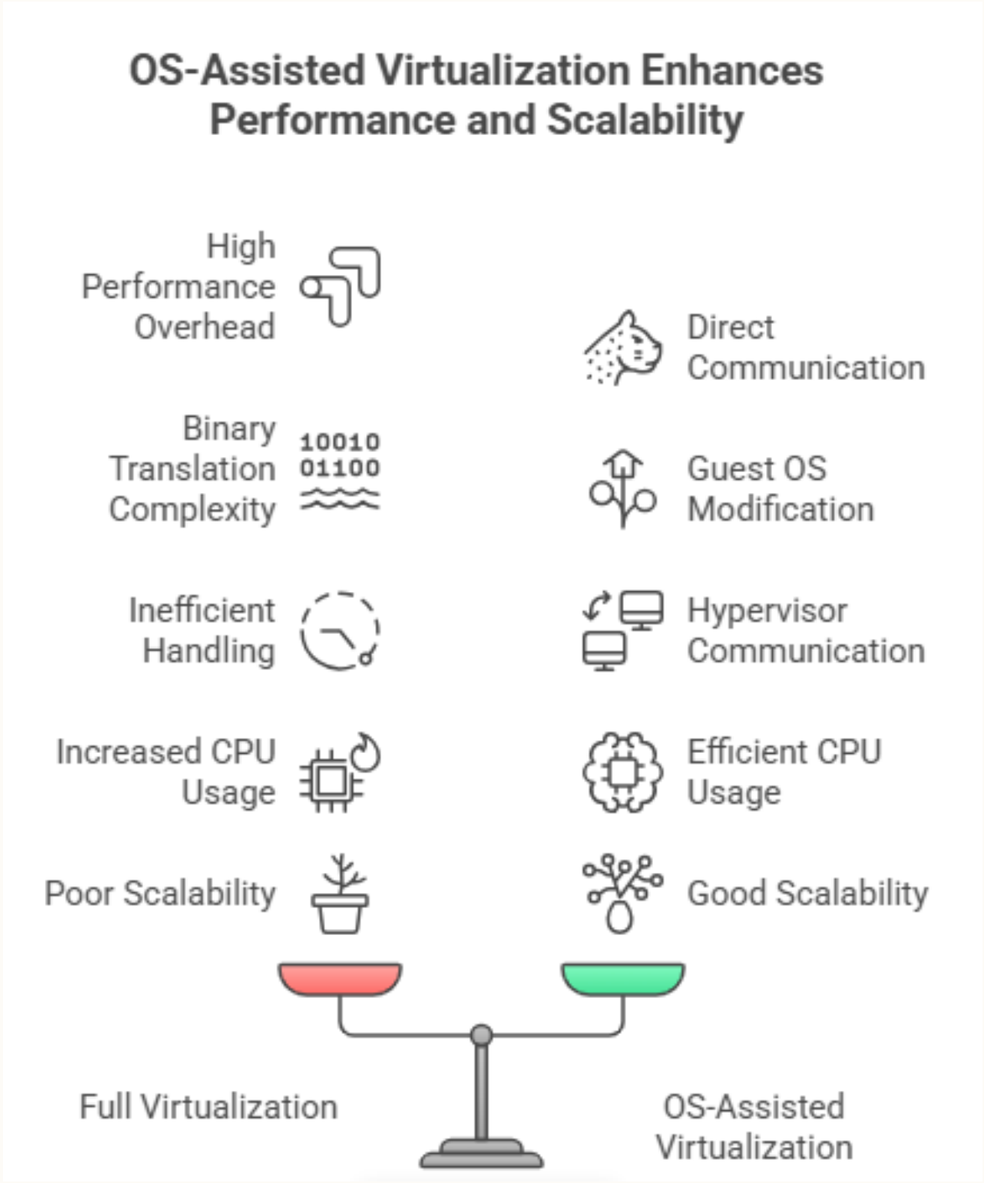
Topic 7:OS assisted /Para virtualization

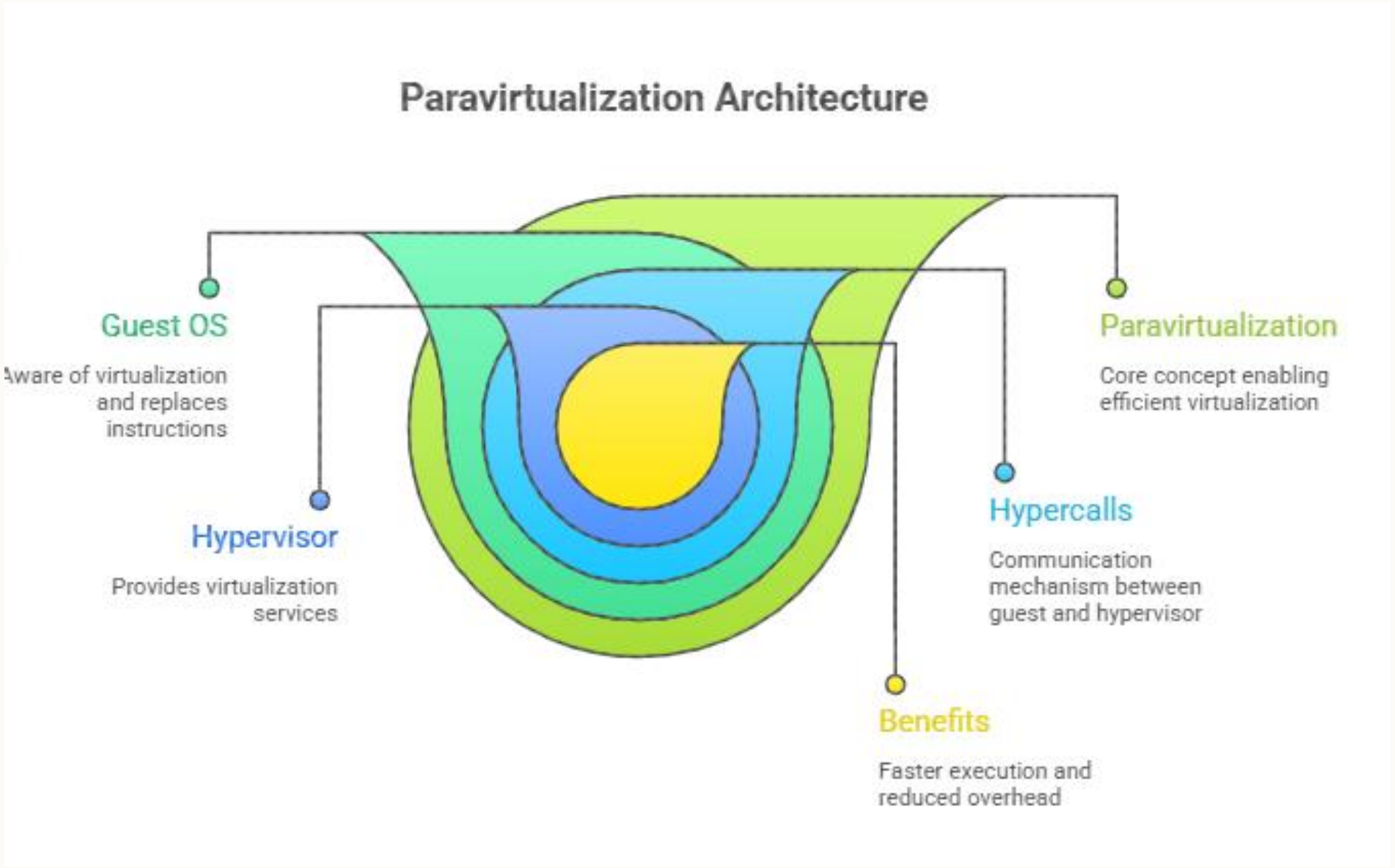
Introduction to Virtualization

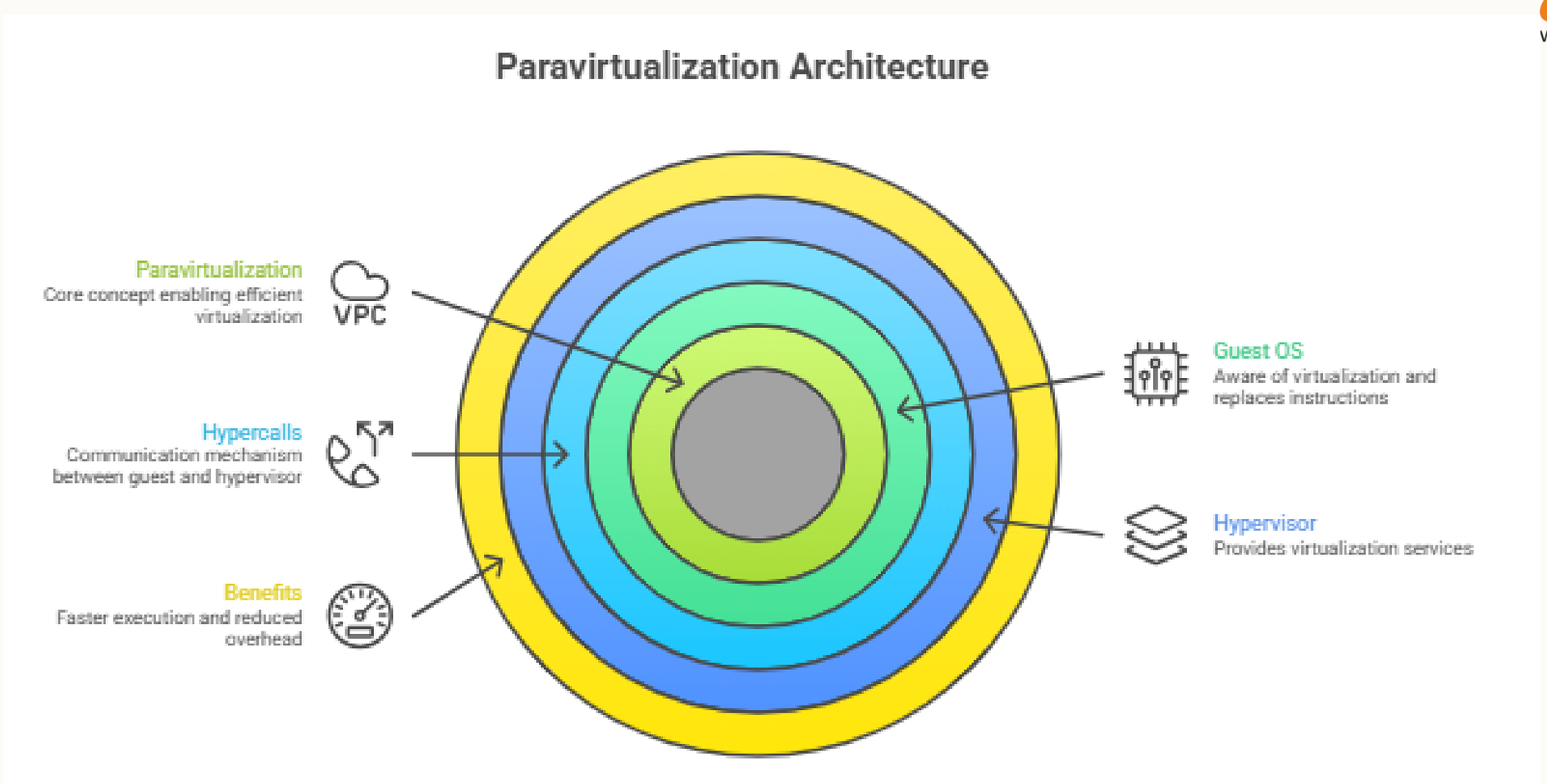


www.snsgroups.com

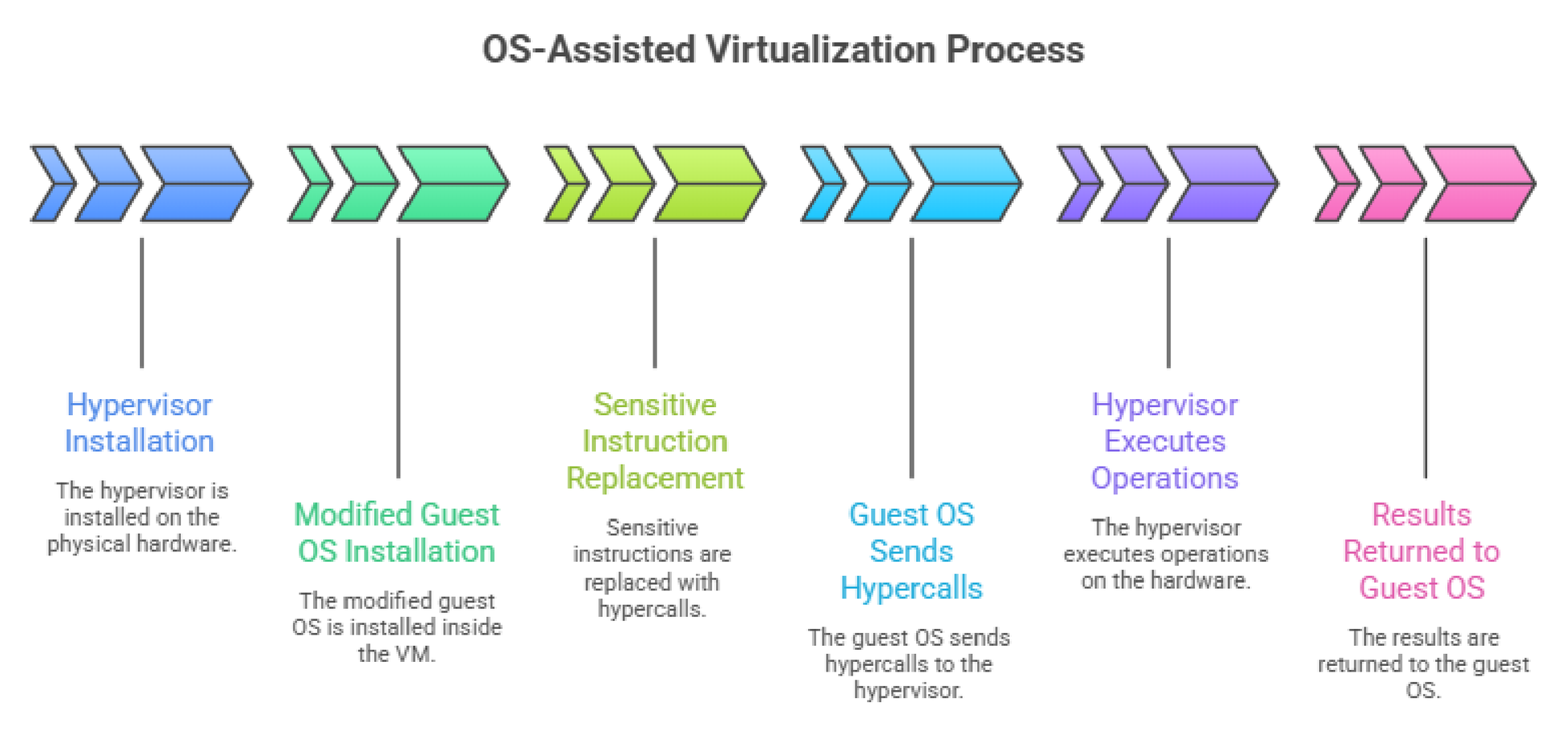



OS-Assisted Virtualization Enhances Performance and Scalability
High Performance Overhead
Direct Communication
Binary Translation Complexity
10010
01100
Guest OS Modification
Inefficient Handling
Hypervisor Communication
Increased CPU Usage
Efficient CPU Usage
Poor Scalability
Good Scalability
Full Virtualization
OS-Assisted Virtualization

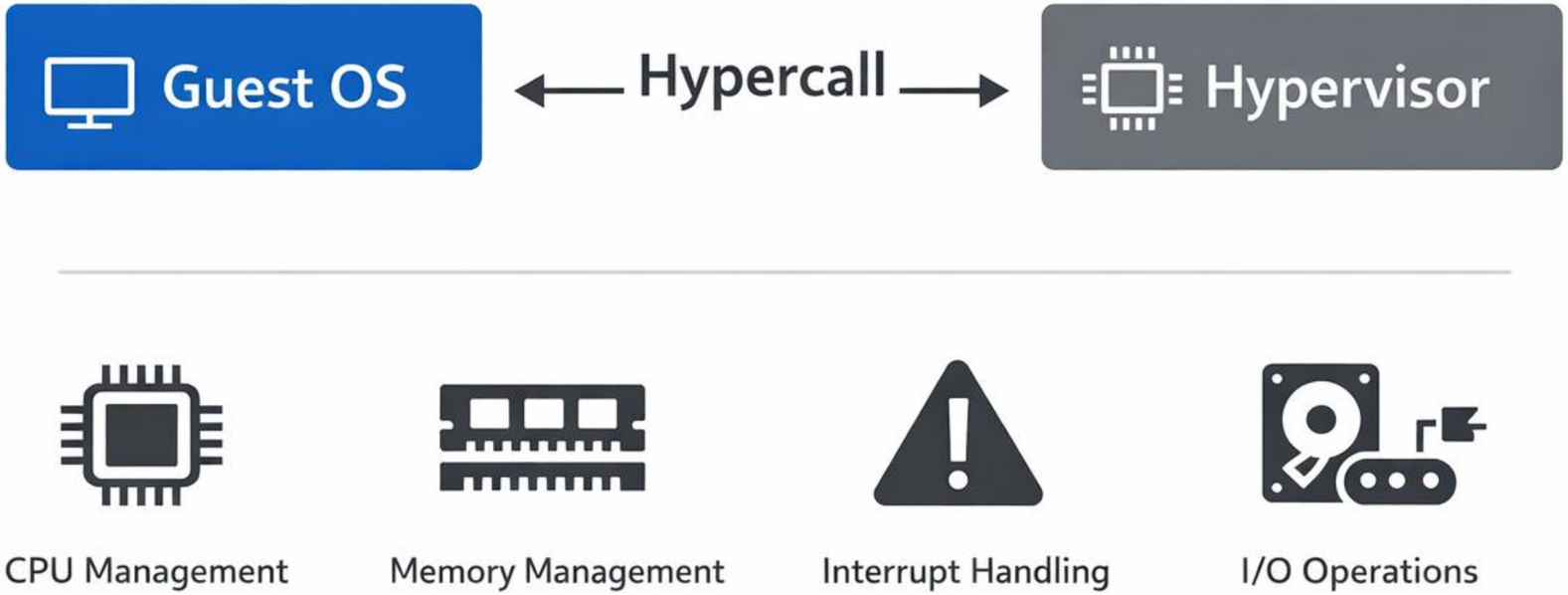

Paravirtualization Architecture
Guest OS
Aware of virtualization and replaces instructions
Hypervisor
Provides virtualization services
Paravirtualization
Core concept enabling efficient virtualization
Hypercalls
Communication mechanism between guest and hypervisor
Benefits
Faster execution and reduced overhead


Paravirtualization Architecture
Paravirtualization
Core concept enabling efficient virtualization
VPC
Hypercalls
Communication mechanism between guest and hypervisor
Benefits
Faster execution and reduced overhead
Guest OS
Aware of virtualization and replaces instructions
Hypervisor
Provides virtualization services

www.snsgroups.com



Role of Hypervisor



OS Modification Requirement

Guest OS
"Must Be Modified"

Kernel source code
must be changed

Privileged instructions
replaced with hypercalls

Supported Operating Systems

Linux

UNIX-like systems

Limited Support

Windows
(closed-source kernel)

Advantages of OS-Assisted Virtualization

✓ Lower performance overhead
✓ Faster execution compared to full virtualization
✓ Efficient CPU utilization
✓ Better memory management
✓ No binary translation required
✓ Simpler hypervisor design

Disadvantages of OS-Assisted Virtualization

- ❌ Requires modification of guest OS
- ❌ Limited OS compatibility
- ❌ Not suitable for proprietary OS (Windows)
- ❌ Less flexible than hardware-assisted virtualization

Problem Statement

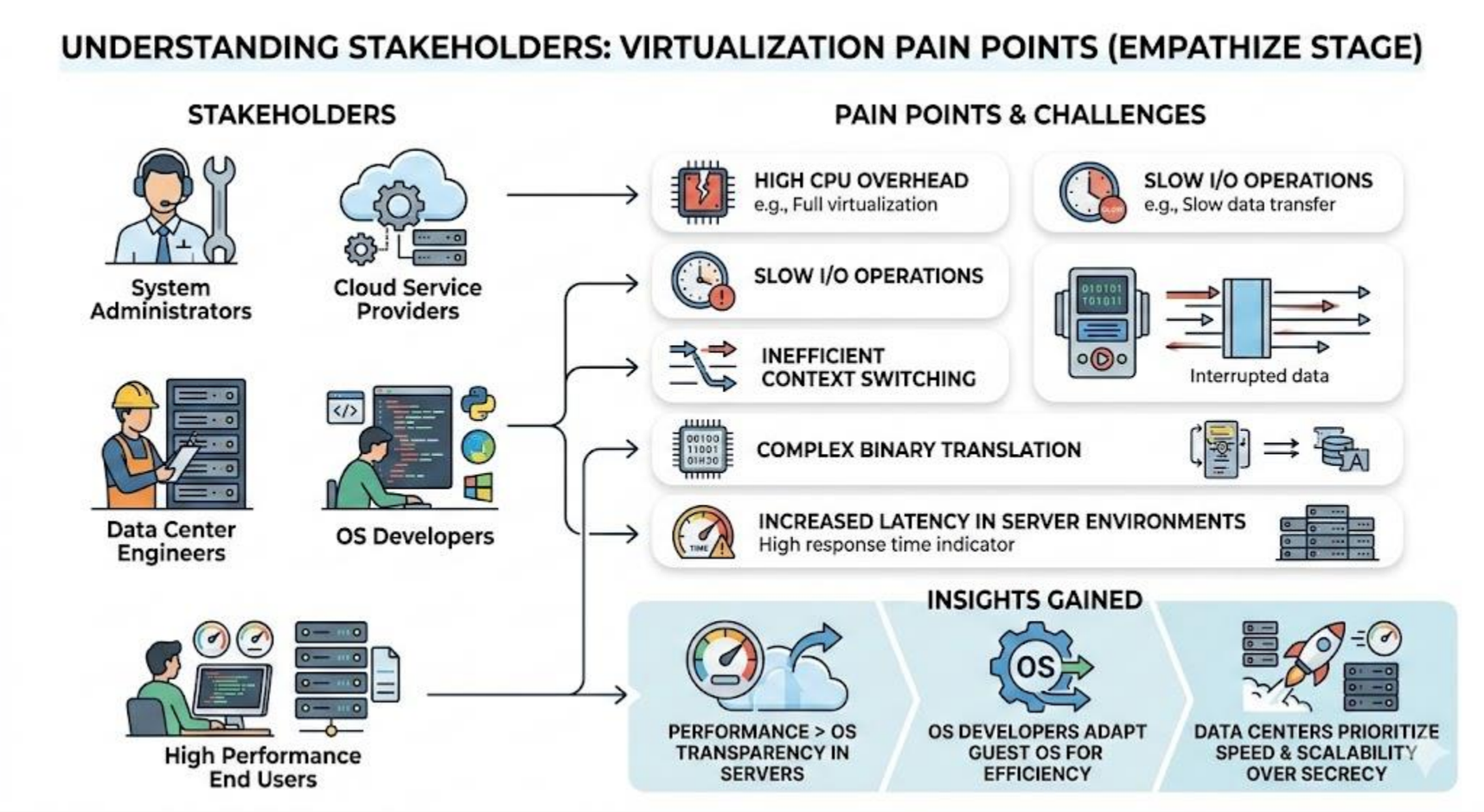
Traditional Full Virtualization

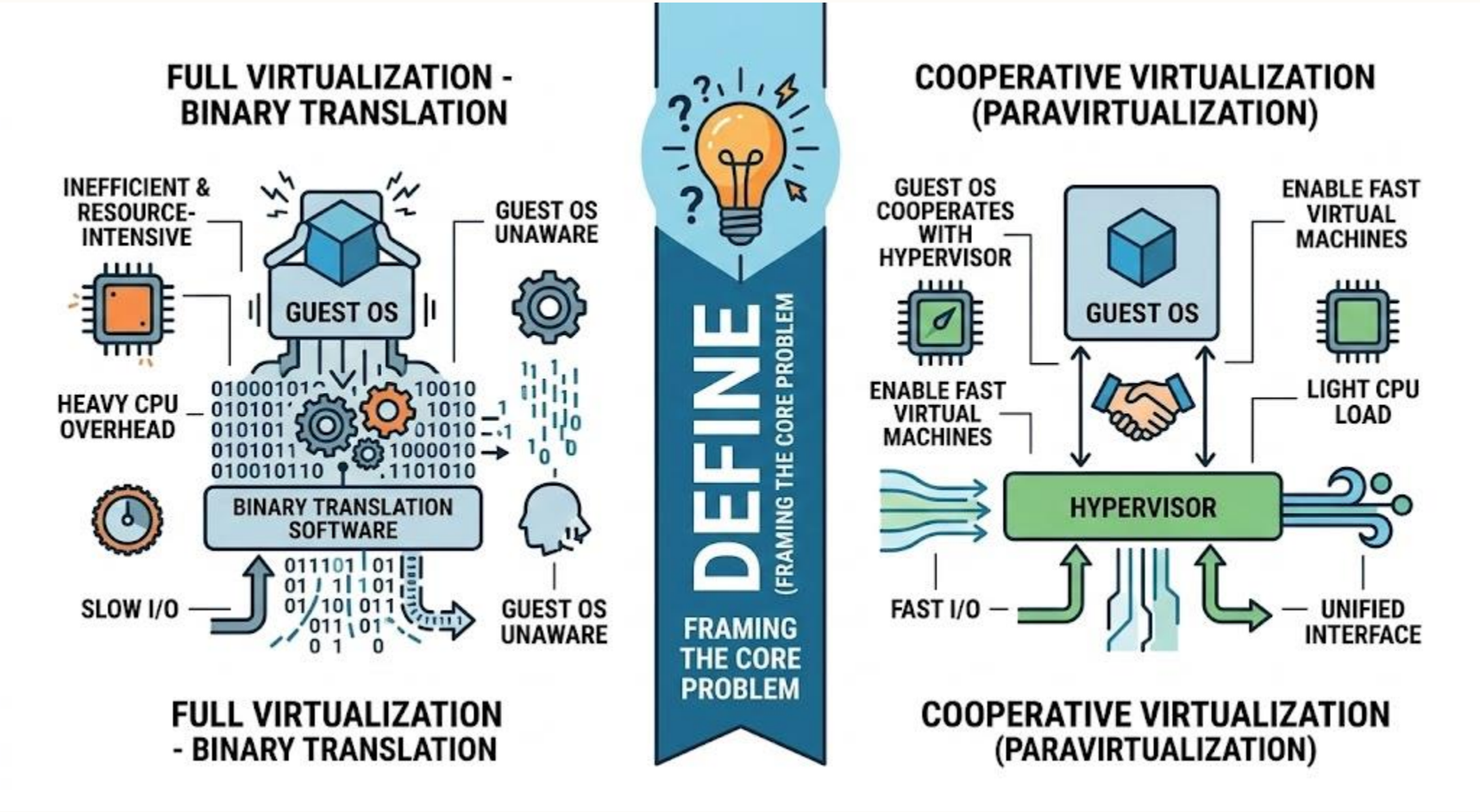
- Guest OS Unaware of Virtualization
- Sensitive Instructions Trapped & Translated By Hypervisor
- Slower Execution & Inefficient CPU Usage

Formal Problem Statement

Reduce Virtualization Overhead & Improve Performance

- Reduce Overhead
- Secure Multi-OS


UNDERSTANDING STAKEHOLDERS: VIRTUALIZATION PAIN POINTS (EMPATHIZE STAGE)
STAKEHOLDERS
PAIN POINTS & CHALLENGES
System Administrators
Cloud Service Providers
Data Center Engineers
OS Developers
High Performance End Users
HIGH CPU OVERHEAD
e.g., Full virtualization
SLOW I/O OPERATIONS
e.g., Slow data transfer
SLOW I/O OPERATIONS
INEFFICIENT CONTEXT SWITCHING
Interrupted data
COMPLEX BINARY TRANSLATION
INCREASED LATENCY IN SERVER ENVIRONMENTS
High response time indicator
INSIGHTS GAINED
PERFORMANCE > OS TRANSPARENCY IN SERVERS
OS DEVELOPERS ADAPT GUEST OS FOR EFFICIENCY
DATA CENTERS PRIORITIZE SPEED & SCALABILITY OVER SECRECY


FULL VIRTUALIZATION - BINARY TRANSLATION
INEFFICIENT & RESOURCE-INTENSIVE
GUEST OS UNAWARE
GUEST OS
HEAVY CPU OVERHEAD
BINARY TRANSLATION SOFTWARE
SLOW I/O
GUEST OS UNAWARE
FULL VIRTUALIZATION - BINARY TRANSLATION
DEFINE
(FRAMING THE CORE PROBLEM
FRAMING THE CORE PROBLEM
COOPERATIVE VIRTUALIZATION (PARAVIRTUALIZATION)
GUEST OS COOPERATES WITH HYPERVISOR
ENABLE FAST VIRTUAL MACHINES
GUEST OS
ENABLE FAST VIRTUAL MACHINES
LIGHT CPU LOAD
HYPERVISOR
FAST I/O
UNIFIED INTERFACE
COOPERATIVE VIRTUALIZATION (PARAVIRTUALIZATION)

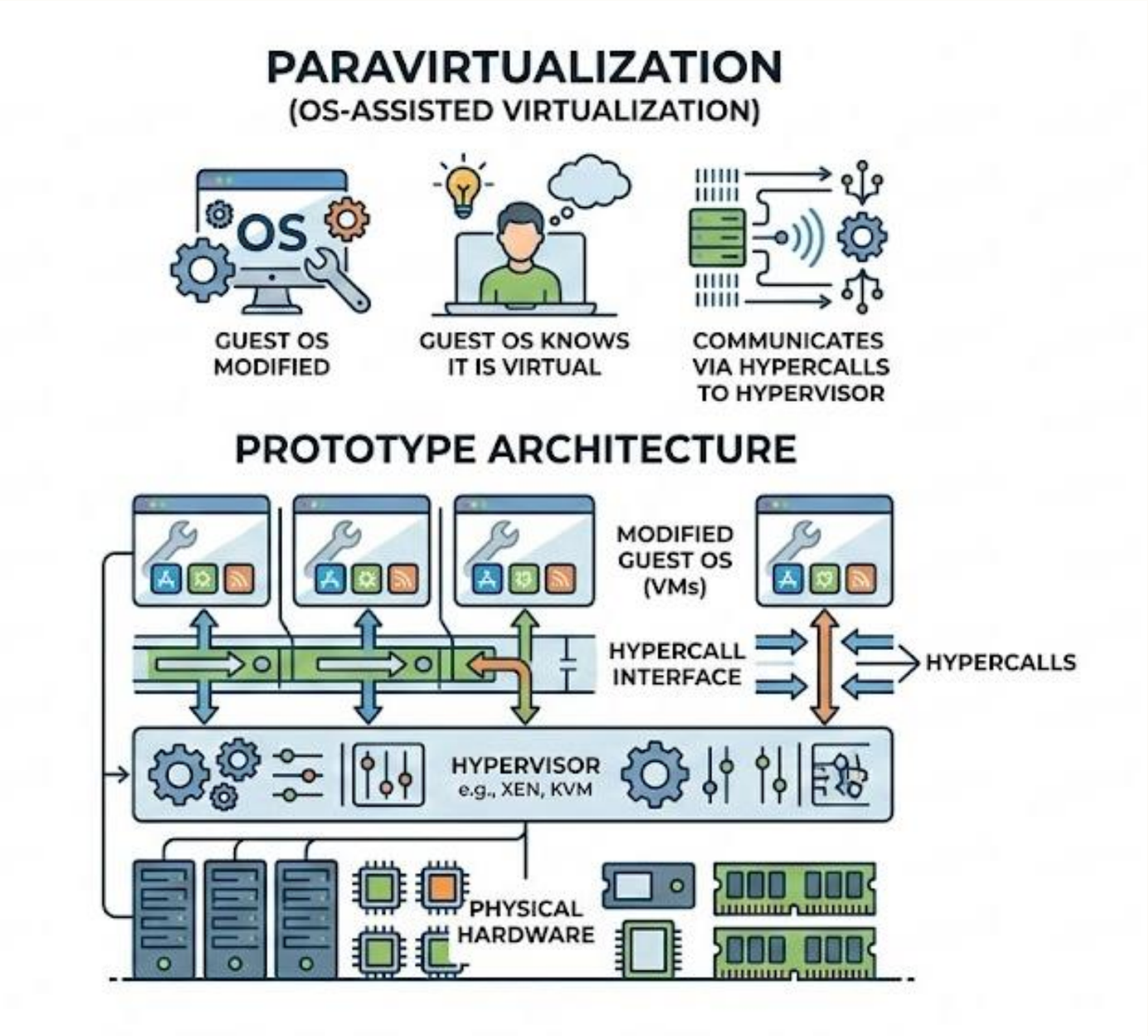
Ideate (Generating Solutions)

Brainstormed Ideas

- Modify guest OS to be virtualization-aware
- Replace privileged instructions with direct hypercalls
- Allow guest OS to communicate directly with hypervisor
- Reduce trap-and-emulate operations
- Optimize I/O using shared drivers

Selected Idea

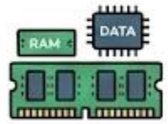
👉 OS-Assisted / Paravirtualization


PARAVIRTUALIZATION
(OS-ASSISTED VIRTUALIZATION)
OS
GUEST OS MODIFIED
GUEST OS KNOWS IT IS VIRTUAL
COMMUNICATES VIA HYPERCALLS TO HYPERVISOR
PROTOTYPE ARCHITECTURE
MODIFIED GUEST OS (VMs)
HYPERCALL INTERFACE
HYPERCALLS
HYPERVISOR
e.g., XEN, KVM
PHYSICAL HARDWARE

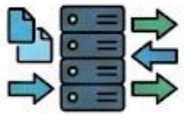
6. TEST (EVALUATION & VALIDATION)

TESTING PARAMETERS

CPU UTILIZATION



MEMORY ACCESS SPEED



I/O PERFORMANCE

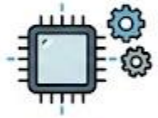
LATENCY

LATENCY

STABILITY

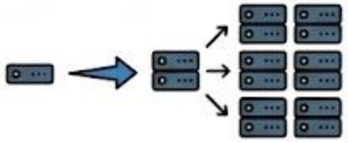
OBSERVATIONS

FASTER EXECUTION



REDUCED CPU OVERHEAD

IMPROVED I/O THROUGHPUT



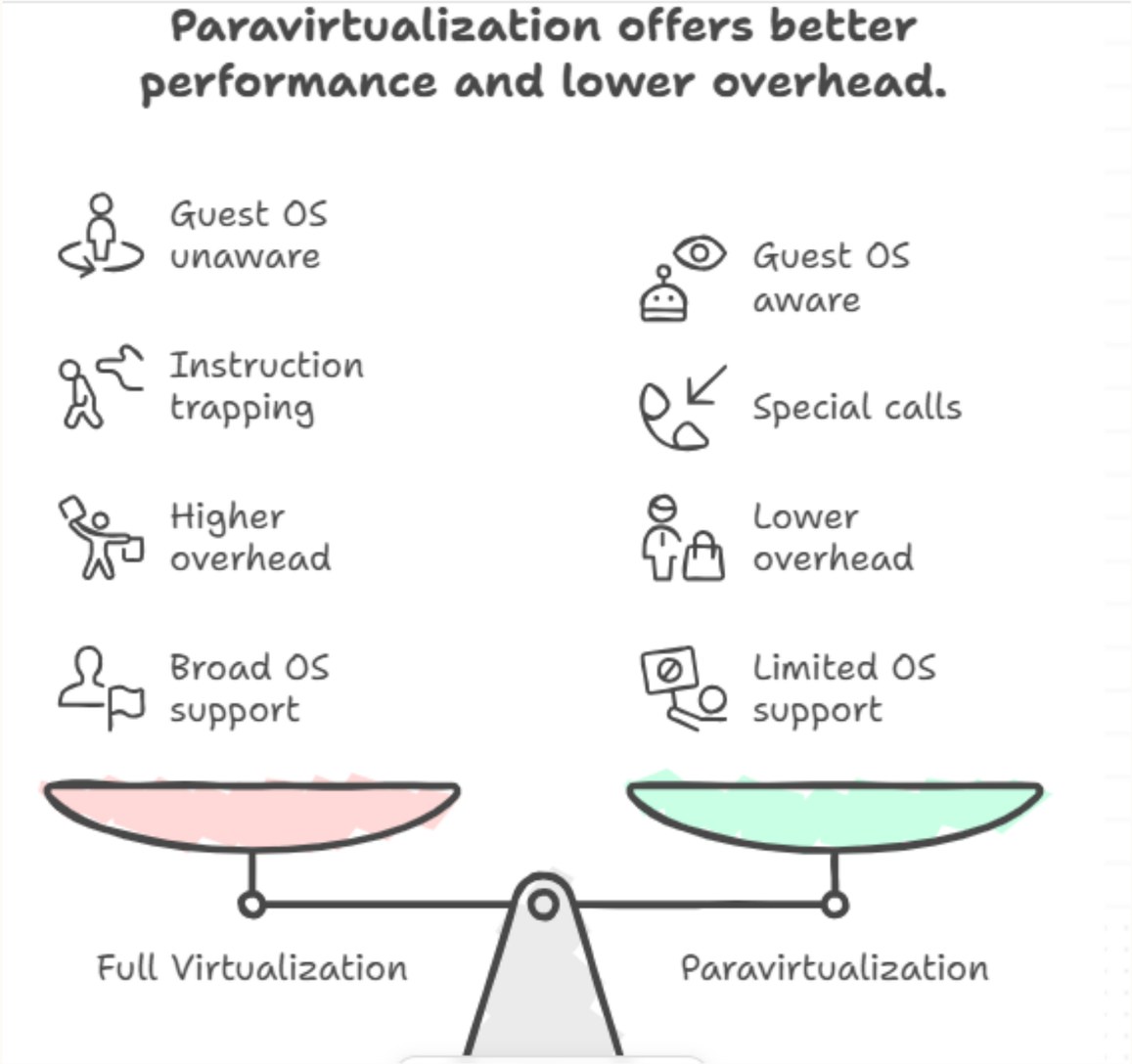
BETTER SCALABILITY

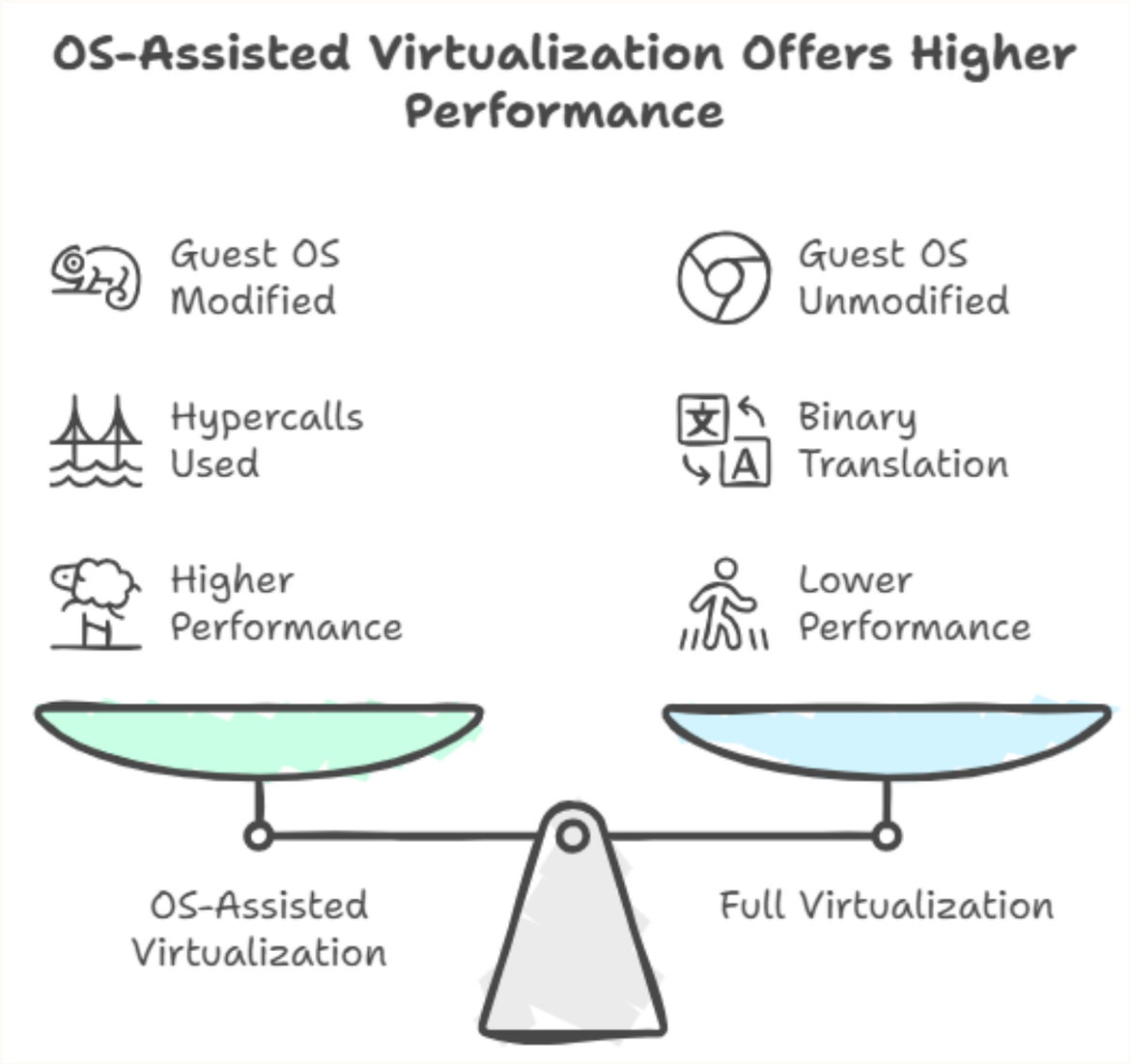
LIMITATIONS FOUND

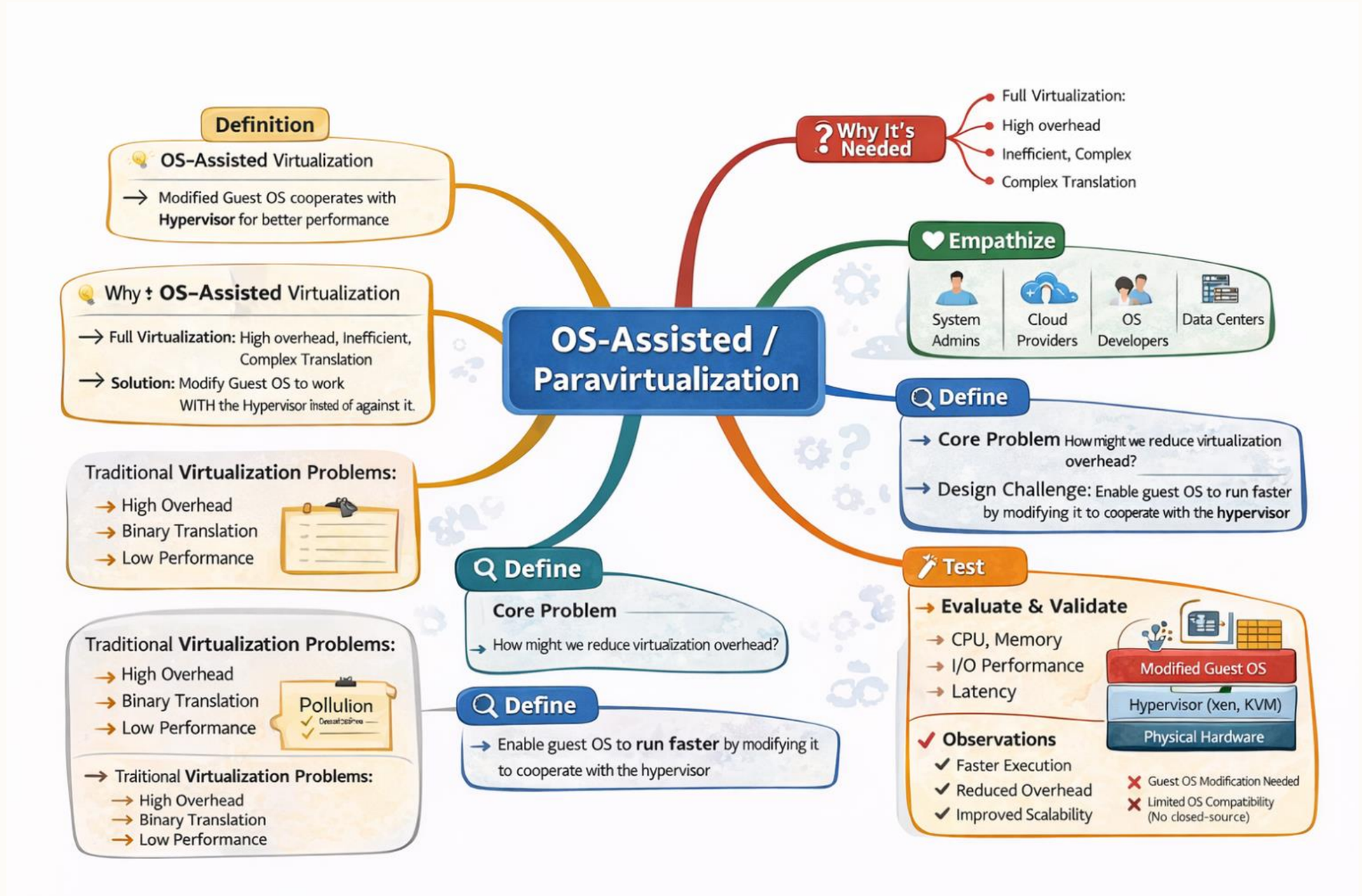
GUEST OS MODIFICATION REQUIRED

CANNOT RUN CLOSED-SOURCE OS EASILY (e.g., OLDER WINDOWS)

LESS FLEXIBILITY THAN FULL VIRTUALIZATION


Paravirtualization offers better performance and lower overhead.
Guest OS unaware
Instruction trapping
Higher overhead
Broad OS support
Guest OS aware
Special calls
Lower overhead
Limited OS support
Full Virtualization
Paravirtualization


OS-Assisted Virtualization Offers Higher Performance
Guest OS Modified
Guest OS Unmodified
Hypercalls Used
Binary Translation
Higher Performance
Lower Performance
OS-Assisted Virtualization
Full Virtualization


Definition
OS-Assisted Virtualization
Modified Guest OS cooperates with Hypervisor for better performance
Why OS-Assisted Virtualization
Full Virtualization: High overhead, Inefficient, Complex Translation
Solution: Modify Guest OS to work WITH the Hypervisor instead of against it.
Traditional Virtualization Problems:
High Overhead
Binary Translation
Low Performance
Traditional Virtualization Problems:
High Overhead
Binary Translation
Low Performance
Pollulion
Traditional Virtualization Problems:
High Overhead
Binary Translation
Low Performance
OS-Assisted / Paravirtualization
Define
Core Problem
How might we reduce virtualization overhead?
Define
Enable guest OS to run faster by modifying it to cooperate with the hypervisor
Why It's Needed
Full Virtualization:
High overhead
Inefficient, Complex
Complex Translation
Empathize
System Admins
Cloud Providers
OS Developers
Data Centers
Define
Core Problem How might we reduce virtualization overhead?
Design Challenge: Enable guest OS to run faster by modifying it to cooperate with the hypervisor
Test
Evaluate & Validate
CPU, Memory
I/O Performance
Latency
Modified Guest OS
Hypervisor (xen, KVM)
Physical Hardware
Observations
Faster Execution
Reduced Overhead
Improved Scalability
Guest OS Modification Needed
Limited OS Compatibility (No closed-source)