1. Summary of Server Components

- **Motherboard (System Board)**: The main circuit board connecting all components, providing communication pathways and power distribution.
- **CPU (Central Processing Unit)**: The processor executing instructions, handling computations, and managing data flow.
- **RAM (Memory)**: Temporary storage for active data and instructions, enabling fast access by the CPU.
- **Storage Drives (HDD/SSD/NVMe)**: Persistent storage for data; HDDs use spinning disks, SSDs use flash memory, and NVMe offers high-speed access via PCIe.
- **RAID Controller (Smart Array)**: Manages multiple drives for redundancy or performance, implementing RAID levels like 0, 1, 5.
- **Power Supply Unit (PSU)**: Converts AC to DC power, supplying stable voltage to components.
- **Network Interface Card (NIC)**: Enables network connectivity, facilitating data transfer over Ethernet or other protocols.
- **Cooling System (Fans and Heat Sinks)**: Dissipates heat from components to prevent overheating, using fans for airflow and heat sinks for thermal transfer.
- **Expansion Slots (PCIe)**: Slots for adding peripheral cards (e.g., GPUs, NICs) to enhance functionality.
- **Chassis (Rack/Tower/Blade)**: The physical enclosure; racks for data centers, towers for standalone servers, blades for compact, modular setups.
- **BIOS/UEFI Firmware**: Firmware initializing hardware during boot and providing a runtime interface for the OS.
- **Backplane**: A board connecting multiple drives or blades, simplifying cabling and enabling hot-swapping.

2. IPMI and iLO: Functions

- **IPMI (Intelligent Platform Management Interface)**: A set of standards for remote server management, allowing monitoring and control (e.g., power, temperature, logs) via a Baseboard Management Controller (BMC).
- **iLO (Integrated Lights-Out)**: HPE's proprietary management technology, providing remote access, monitoring, and control (e.g., power, firmware updates, virtual media) via a dedicated chip.

3. Relation of IPMI/iLO to BIOS/UEFI

- IPMI/iLO operate independently of BIOS/UEFI but interact during boot for tasks like remote configuration or firmware updates. The BMC (for IPMI) or iLO chip can access BIOS/UEFI settings, monitor hardware, or initiate reboots, complementing firmware's role in hardware initialization.

4. CPU Sockets: Purpose

- **CPU Sockets**: Physical connectors on the motherboard that house the CPU, ensuring proper alignment and electrical connectivity. They match specific CPU types (e.g., LGA, PGA) to support processing needs and enable upgrades.

5. Why Pseudo File System in Linux?

- Introduced to align with Linux's philosophy of treating everything as a file, pseudo file systems (e.g., `/proc`, `/sys`) provide a file-based interface to kernel and hardware data, simplifying access and management without physical storage.

6. Pseudo File System vs. Normal File System

- **Pseudo File System**:
  - Virtual, in-memory representation of kernel/hardware data.
  - No physical storage; dynamically generated by the kernel.
  - Examples: `/proc`, `/sys`, `/dev`.
- **Normal File System**:
  - Stores data on physical media (e.g., HDD, SSD).
  - Persistent across reboots.
  - Examples: ext4, NTFS, XFS.

7. Information in /sys/ Directory

- The `/sys/` directory exposes kernel and device information via a pseudo file system (`sysfs`). It includes:
  - **power**: Power management settings.
  - **block**: Block device info (e.g., disks).
  - **bus**: Bus types (e.g., PCI, USB).
  - **class**: Device classes (e.g., network, input).
  - **dev**: Device mappings.
  - **devices**: Hardware device details.
  - **firmware**: Firmware attributes.
  - **fs**: File system parameters.
  - **hypervisor**: Virtualization info.
  - **kernel**: Kernel parameters.
  - **module**: Loaded kernel modules.

8. DMA (Direct Memory Access) in Linux

- **DMA**: Allows devices to transfer data directly to/from RAM, bypassing the CPU, improving efficiency.
- **Use Case in Linux**: Used by devices like disk controllers, NICs, or GPUs to handle large data transfers (e.g., disk I/O, network packets) without CPU overhead.

9. lsblk Command Internals

- **lsblk**: Lists block devices (e.g., disks, partitions) by reading `/sys/block` and `/proc/partitions`, querying device attributes like size, mount points, and hierarchy.
- **Similar Commands**:

- **lsusb**: Reads `/sys/bus/usb` to list USB devices.
- **lspci**: Queries `/sys/bus/pci` for PCI devices.
- **lshw**: Aggregates data from `/sys`, `/proc`, and other sources for comprehensive hardware details.
- All rely on kernel interfaces but target different subsystems.

10. Simulate Shutdown via /sys

- Write `1` to `/sys/power/state` (e.g., `echo 1 > /sys/power/state`) to trigger a shutdown or suspend, as this interface controls power management states. Requires root privileges.

11. Types of Kernels

- **Monolithic Kernel**:
  - All OS services (e.g., file systems, drivers) run in kernel space.
  - **Advantages**: High performance, direct hardware access.
  - **Disadvantages**: Large, complex, less modular; crashes affect entire system.
- **Microkernel**:
  - Minimal kernel; services (e.g., drivers) run in user space.
  - **Advantages**: Modular, stable, easier to maintain.
  - **Disadvantages**: Slower due to inter-process communication.
- **Hybrid Kernel**:
  - Combines monolithic and microkernel traits; some services in kernel space, others in user space.
  - **Advantages**: Balances performance and modularity.
  - **Disadvantages**: Complex design, potential stability issues.

12. Why First Sector for MBR?

- The first 512 bytes (first sector) of a disk are used for the Master Boot Record (MBR) because it's a standardized location for bootloaders, ensuring compatibility and quick access during the boot process.

13. How MBR Locates GRUB/Bootloader

- The MBR contains a partition table and boot code. The boot code points to the bootloader (e.g., GRUB), typically located in a designated partition or the next sectors, using pointers to its disk address.

14. .efi Files in Boot Process

- **.efi Files**: Executable files in UEFI systems, written in the EFI bytecode format.
- **Role**: Act as bootloaders or EFI applications (e.g., GRUB, OS loaders) executed by UEFI firmware to initialize the OS.

15. ESP (EFI System Partition) in UEFI

- **ESP**: A FAT-formatted partition storing UEFI bootloaders, kernels, and drivers.
- **Use**: UEFI firmware reads .efi files from ESP to start the boot process, ensuring compatibility across systems.

16. Explanation of /etc/grub/grub.conf Section

- **menuentry 'Ubuntu'**: Defines a GRUB boot menu entry for Ubuntu.
- **--class**: Tags for menu styling (e.g., `ubuntu`, `gnu-linux`).
- **$menuentry_id_option**: Unique identifier for the entry.
- **recordfail**: Tracks boot failures to trigger recovery if needed.
- **load_video**: Loads video drivers for graphical boot.
- **gfxmode $linux_gfx_mode**: Sets graphical mode for boot.
- **insmod gzio, xzio, lzopio, part_gpt, ext2**: Loads GRUB modules for compression, GPT partitioning, and ext2 file system.
- **set root='hd0,gpt2'**: Specifies the root device (second GPT partition on first disk).
- **search --no-floppy --fs-uuid ...**: Locates the root partition by UUID, with hints for BIOS/EFI modes.
- **linux /vmlinuz-5.4.0-65-generic ...**: Loads the kernel with parameters (e.g., root device, read-only mode).
- **initrd /initrd.img-5.4.0-65-generic**: Loads the initial RAM disk for kernel boot.