University of St.Thomas of Mozambique

Business School

Subject: Parallel Computing

Exam

Student: Jennifer Manhice

Code: 202100894

Class: 4P7CS/3P5CS

Lecturer: Mr Lars Lemos

July 04st 2024

1. i) Ans:

Computing Concepts and Parts to Optimise

1. **Storage**:
    - The 900 million people represent the total data set that needs to be processed. In parallel computing, this is akin to the total data that needs to be distributed across multiple processing units.
2. **RAM**:
    - Each train can carry 1000 people, similar to how RAM can handle a limited amount of data at a time. This determines the size of the chunk of data that can be processed simultaneously.
3. **Bus**:
    - The 4 railways are like the data buses that transport data between the storage and the CPU. Optimising the bus width (number of railways) can improve data transfer rates.
4. **CPU**:
    - The metro stations are the processing units where the data is processed. More efficient processing units or better scheduling can improve overall performance.
5. **Processes**:
    - The 10 trains going and 10 returning represent the concurrent tasks being processed. Efficiently managing these processes and minimising idle times can enhance throughput.

Optimization Parts:

- **Increase Bus Width**: If possible, adding more railways (data buses) can increase the data transfer rate, reducing the time spent moving data between storage and processing units.
- **Improve Scheduling**: Optimising the schedule of trains (processes) to ensure that no train (process) is idle can improve efficiency. This could mean better load balancing and reducing bottlenecks.
- **Enhance Processing Power**: Upgrading the metro stations (CPUs) to process data more quickly can reduce overall processing time.
- **Expand RAM Capacity**: If trains could carry more people (increasing RAM), larger chunks of data could be processed at once, reducing the number of trips needed.

ii)Ans:

Worst Scenario:

- **Bottlenecks and Idle Time:** If the railways (data buses) are congested, and there are delays in transferring people (data) to and from the metro stations (CPUs), the process

becomes inefficient. This is akin to having limited bandwidth in a computer system, causing delays.

- **Insufficient RAM:** If the trains (RAM) are too small to handle the load efficiently, many trips will be required, increasing the overall time.
- **Unbalanced Load:** If some trains (processes) are idle while others are overloaded, it leads to inefficient use of resources. This can happen if the scheduling is not optimised

.

Best Scenario:

- **Optimised Data Transfer:** If the railways (data buses) are fully utilised without congestion, data transfer between storage and processing units is seamless.
- **Adequate RAM:** If each train (RAM) can carry a large enough chunk of data, fewer trips are required, reducing the overall processing time.
- **Balanced Load:** Efficient scheduling ensures that all trains (processes) are evenly loaded, minimising idle times and maximising resource usage.
- **Enhanced Processing Power:** If the metro stations (CPUs) can process data quickly, the overall time to complete the processing is reduced.

2. Ans:

1. **Data Storage:**

Internal Storage, ROM Capacity (NAND Flash Memory):
This is the main storage for the phone, used to store the operating system, apps, and user data. NAND flash memory is fast and non-volatile, ensuring quick access and data retention without power.

RAM (Random Access Memory):
RAM is used for temporary data storage that the CPU needs to access quickly. The amount and speed of RAM directly affect the phone's ability to handle multiple tasks simultaneously and efficiently.

Storage Controller:
This chip manages the flow of data between the CPU and the internal storage. A good storage controller optimises read and write speeds and improves overall storage efficiency.

2. **Data Transfer**

CPU (Central Processing Unit):
The CPU processes instructions and handles data transfer tasks. A multi-core, high-speed CPU can process more data simultaneously, improving overall performance.

Internal Buses:
UFS (Universal Flash Storage): This is a high-speed interface used for internal data transfer between the CPU, RAM, and NAND flash memory. UFS provides faster data transfer rates compared to older technologies like eMMC.
PCIe (Peripheral Component Interconnect Express): Some high-end phones use PCIe for even faster data transfer rates.

USB Interface:
USB-C: This port is used for external data transfer and charging. USB-C supports high-speed data transfer and is reversible, making it more user-friendly.

Wireless Communication Chips:
Wi-Fi Chip: Handles connections to Wi-Fi networks. The efficiency of data transfer over Wi-Fi depends on the Wi-Fi version supported by the chip (e.g., Wi-Fi 5, Wi-Fi 6).
Bluetooth Chip: Manages short-range wireless communication with other devices. Bluetooth 5.0 and later versions offer improved data transfer speeds and range.

3. **Connectivity**

Modem:
The modem connects the phone to cellular networks. Modern phones use advanced modems that support multiple generations of network technology (e.g., 4G LTE, 5G) for faster and more reliable connectivity.

Antenna System:
The antenna system is crucial for signal reception and transmission. Well-designed antennas enhance connectivity and data transfer speeds by improving signal strength and quality.

Network Interface Chips:
These chips handle the phone's connections to various networks, including cellular, Wi-Fi, and Bluetooth. Advanced network interface chips support faster and more efficient data connections.

SIM Card and eSIM:
These components manage subscriber identity and connect the phone to the mobile network. An eSIM can switch between carriers without needing a physical SIM card, offering more flexibility and potentially better connectivity.

Additional Factors
Battery and Power Management:
Efficient power management systems ensure that components like the CPU, RAM, and network interfaces operate optimally without draining the battery too quickly. This includes dynamic voltage and frequency scaling (DVFS) to adjust power and performance based on current needs.

Thermal Management:
Effective thermal management prevents overheating, which can throttle performance. Components such as heat pipes, thermal paste, and cooling systems help maintain optimal temperatures and ensure consistent performance.

3. Ans:

Processing 1 billion transactions on a server and a phone involves considering their hardware capabilities, including CPU performance, memory, storage, and network bandwidth. Here's a comparison of how each device might handle such a task, along with the relevant metrics:

Server

1. **CPU**:
    - **Architecture**: Typically, servers use multi-core, high-performance CPUs like Intel Xeon or AMD EPYC.
    - **Clock Speed**: Around 2.0 - 3.5 GHz per core.
    - **Core Count**: Often 16 to 64 cores or more.
2. **Memory (RAM)**:
    - **Capacity**: Servers can have 128 GB to several TB of RAM.
    - **Speed**: Generally DDR4 or DDR5 with high bandwidth.
3. **Storage**:
    - **Type**: SSDs (NVMe) with high IOPS (Input/Output Operations Per Second).
    - **Capacity**: Can range from TBs to PBs.
4. **Network Bandwidth**:
    - **Speed**: Typically 10 Gbps to 100 Gbps.
5. **Metrics**:
    - **Transaction Processing Rate**: A powerful server can process tens of thousands to hundreds of thousands of transactions per second (TPS).
    - **Latency**: Generally low, measured in milliseconds.

Phone

1. **CPU**:
    - **Architecture**: ARM-based processors, e.g., Qualcomm Snapdragon or Apple A-series.
    - **Clock Speed**: Around 1.8 - 3.0 GHz per core.
    - **Core Count**: Typically 4 to 8 cores.
2. **Memory (RAM)**:
    - **Capacity**: 4 GB to 16 GB.
    - **Speed**: LPDDR4 or LPDDR5.
3. **Storage**:
    - **Type**: Flash storage.

    - **Capacity**: Ranges from 64 GB to 1 TB.
4. **Network Bandwidth**:
    - **Speed**: Up to 5G speeds (theoretically several Gbps).
5. **Metrics**:
    - **Transaction Processing Rate**: A high-end phone can process hundreds to a few thousand TPS.
    - **Latency**: Higher compared to servers, generally in tens of milliseconds.

Example Metrics Calculation

To process 1 billion transactions, let's calculate the time required based on hypothetical TPS values:

Server

- **Assumed Transaction Per Second (TPS)**: 100,000 TPS.
- **Total Transactions**: 1,000,000,000 transactions.
- **Time Required**:

  Time(seconds)= 1,000,000,000/100,000= 10 000 seconds

  Time(hour)= 10 000/3600 = 2.78 hours

Time required for a server is 2.78 hours

Phone

- **Assumed TPS**: 1,000 TPS.
- **Total Transactions**: 1,000,000,000 transactions.
- **Time Required**:

  Time (seconds)= 1,000,000,000/1,000=1,000,000 seconds

  Time (hour)= 1,000,000/3600= 277.78 hours

Server Hardware Capabilities

1. **Multi-Core CPUs**:
    - **Parallel Processing**: Servers use multi-core CPUs to handle multiple transactions simultaneously. Each core can execute a thread, allowing the server to process many transactions in parallel.
2. **Large Memory (RAM)**:
    - **Fast Data Access**: Sufficient RAM ensures that the data required for transactions is quickly accessible, reducing latency.
    - **Caching**: Frequently accessed data can be stored in RAM, speeding up repeated transactions.

3. **High-Speed Storage**:
    - **Quick Read/Write**: NVMe SSDs provide high IOPS, allowing the server to quickly read from and write to storage.
    - **Data Management**: Efficient storage management and RAID configurations help in managing large volumes of data securely and reliably.
4. **High Network Bandwidth**:
    - **Fast Data Transfer**: High network speeds (10 Gbps to 100 Gbps) allow for rapid communication between the server and external systems or users.
    - **Load Balancing**: Network load balancing distributes incoming transaction requests evenly across server resources.

Server Software Capabilities

1. **Operating System**:
    - **Resource Management**: The OS efficiently manages CPU, memory, and I/O resources to ensure smooth operation.
    - **Concurrency**: Advanced operating systems provide robust support for multithreading and multiprocessing.
2. **Database Management System (DBMS)**:
    - **Transaction Management**: DBMSs like MySQL, PostgreSQL, or Oracle handle transactions with ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring reliability and correctness.
    - **Query Optimization**: Efficient query execution plans reduce the time required to process each transaction.
3. **Middleware and Application Servers**:
    - **Scalability**: Middleware can distribute transaction processing across multiple servers if needed.
    - **Load Balancing**: Application servers balance the load across available resources to prevent bottlenecks.
4. **Task Scheduling and Thread Management**:
    - **Thread Pools**: Servers use thread pools to manage a large number of concurrent transaction processing threads.
    - **Task Queues**: Tasks are placed in queues and processed by available threads, ensuring efficient resource use.

Example Workflow for Processing Transactions

1. **Request Handling**:
    - Incoming transaction requests are received via the server's network interface.
    - A load balancer distributes the requests to different processing nodes or threads.
2. **Transaction Processing**:
    - Each transaction request is assigned to a thread.

    - The thread executes the transaction logic, which may involve reading/writing to a database, performing calculations, or interacting with other services.
3. **Database Operations**:
    - The server interacts with the DBMS to execute database operations.
    - Efficient indexing, query optimization, and caching are used to speed up data access and manipulation.
4. **Response Generation**:
    - Once a transaction is processed, the result is generated and sent back to the client.
    - The server logs the transaction details for auditing and monitoring purposes.
5. **Concurrency Control**:
    - Concurrency control mechanisms ensure that transactions are executed without conflicts.
    - Techniques like locking, timestamps, and versioning help maintain data integrity.
6. **Monitoring and Scaling**:
    - Continuous monitoring of performance metrics (e.g., TPS, CPU usage, memory usage) helps in identifying bottlenecks.
    - If needed, the server can scale horizontally (adding more servers) or vertically (upgrading server hardware) to handle increased load.

Performance Optimization

1. **Load Balancing**:
    - Distributing transactions evenly across multiple servers or threads prevents any single resource from becoming a bottleneck.
2. **Caching**:
    - Implementing caching strategies (e.g., in-memory caches like Redis or Memcached) reduces the need for repeated database access.
3. **Database Indexing**:
    - Proper indexing of database tables speeds up query execution, reducing transaction processing time.
4. **Efficient Code**:
    - Writing efficient, optimised code ensures that transaction processing logic is executed quickly.