Computer Architecture - Week 9

Khaula Molapo

1.

A multiprocessor design for a cloud server allows several processors to work together to handle many tasks at the same time. In this system, each processor, or CPU core, can process a different user request or data task. This design helps improve the overall speed and performance of the cloud system, especially when thousands of users connect at once. The cloud server uses shared memory and high-speed communication channels so that processors can quickly exchange data. It also includes a load balancer, which ensures that no single processor becomes overloaded while others remain idle. By distributing work evenly, the system becomes more reliable and efficient. If one processor fails, others can take over its tasks, keeping the system running smoothly. Multiprocessor cloud servers are used by big companies like Google, Amazon, and Microsoft to support their cloud computing services. This design increases processing power, reduces response time, and allows flexible scalability for future growth.

2.

NUMA, which stands for Non-Uniform Memory Access, is a computer memory design used in multiprocessor systems. In a NUMA system, each processor has its own local memory that it can access faster than memory connected to another processor. This helps reduce delays and improves performance when running many tasks at once. For example, if a processor can work mostly with its local memory, it doesn't need to wait for data from other processors, which makes the whole system faster. NUMA is used in large servers, data centers, and high-performance computing environments where speed and efficiency are very important. However, if data is not managed properly, performance can drop because accessing remote memory takes more time. Modern operating systems and processors are designed to optimize NUMA systems automatically, ensuring that tasks and memory are placed where they will work the fastest.

3.

In Visual Studio Code, OpenMP can be used to simulate a parallel task using C or C++. OpenMP allows multiple threads to run different parts of a program at the same time. For example, a loop that processes large amounts of data can be divided among several threads, each handled by a different processor core. This reduces the total execution time and shows how parallel computing improves performance. To test this, I wrote a simple C program that uses #pragma omp parallel for to split the loop into multiple threads. When I ran it, the program finished much faster than the normal single-thread version. This simulation helped me understand how OpenMP makes use of all available cores. The reflection is that OpenMP is very useful for speeding up programs that can be broken into smaller tasks, especially in high-performance or scientific computing environments.

4.

The designed multiprocessor architecture includes several processors connected through a shared bus and memory system. Each processor has its own cache and local memory to improve access speed. The system allows all processors to communicate and share data efficiently, which makes it perfect for multitasking environments like data servers or research simulations. The main purpose of this design is to allow the system to handle many processes at the same time without slowing down. When one processor is busy, others can take on different tasks, leading to balanced workload distribution. This increases reliability because even if one processor stops working, the system can still continue running using the others. The architecture also supports scalability, meaning more processors can be added in the future as the demand for computing power grows. Overall, this design improves speed, performance, and stability for complex IT systems.