Compute FLOPs

1. Given a processor with a clock speed of 3.2 GHz and 4 cores, calculate the theoretical peak FLOPS. Assume that a base level of performance where each core can perform one floating-point operation per clock cycle.
2. A CPU has a clock speed of 2.5 GHz and 8 cores. Assuming each core can perform 4 double-precision floating-point operations per clock cycle, what is the peak FLOPS for this CPU?
3. In the problem 2, if single precision floating-point operations are used, what is the peak FLOPS for the CPU?
4. A supercomputer cluster has 10,000 CPU cores, and each core operates at a clock speed of 3.2 GHz. If each core can perform 8 gigaflops (GFLOPS), what is the peak FLOPS for this cluster?
5. A supercomputer consists of 5,000 processing nodes, each with a clock speed of 2.8 GHz. If each node can perform 4 teraflops (TFLOPS), what is the peak FLOPS for this supercomputer?
6. A supercomputer consisting of 442 processing nodes, where each node has 2x20 Intel 6148 cores, and each core operates at a clock speed of 2.4 Ghz, calculate the peak TFLOPS for this supercomputer? each core can perform 32 double-precision floating-point operations per clock cycle

Compute speed and efficiency

1. A parallel algorithm executes on 8 cores in 120 seconds. If the same algorithm runs sequentially in 800 seconds, find the speedup and efficiency.
2. In a cluster, a simulation task is divided among 16 nodes, and it takes 120 seconds to complete. If you increase the number of nodes to 32 and the time reduces to 80 seconds, compute the strong scaling efficiency.
3. In a parallel application, 90% of the code can be parallelized. Calculate the maximum speedup that can be achieved regardless of the number of processors.
4. A simulation program can parallelize 90% of its workload. If it takes 100 seconds to run sequentially and 10 seconds to run on 20 processors, compute the speedup on 20 processors. Is the provided problem statement accurate?

Cost of a computer depends on,

1. CPU: CPUs come in various price ranges, from budget options to high-end processors. High-performance CPUs with larger cache sizes tend to be more expensive.
2. Cache Memory: The cost of cache memory is typically included in the CPU's price, but CPUs with larger and faster caches are costly.
3. Main Memory (RAM): The cost of RAM depends on its capacity, speed, and type (e.g., DDR4, DDR5). More RAM and faster speeds generally result in higher costs.
4. Storage: The cost of storage devices (e.g., SSDs, HDDs) depends on their capacity and technology. SSDs, which are faster than HDDs, can be more expensive on a per-gigabyte basis.
5. Motherboard: Motherboards with support for advanced cache management and multiple memory slots might cost more than basic models.
6. Memory Controller and Bus System: These components are integrated into the motherboard and CPU, so their cost is typically part of the CPU and motherboard expenses.
7. Cache Management Algorithms and Coherency Protocols: These software components are usually included in the system's design and don't have separate costs.
8. Cooling and Power Supply: High-performance systems might require more robust cooling solutions and power supplies, which can add to the overall cost.
9. Graphics Card (if applicable): High-end graphics cards can be costly, especially for gaming or professional workloads.
10. Case and Peripherals: The cost of the computer case and peripherals (monitor, keyboard, mouse, etc.) is separate from the core components but contributes to the overall cost.
11. Operating System and Software: The cost of the operating system and any additional software licenses must be considered.
12. Labor and Assembly: If you're not building the system yourself, there may be labor costs associated with assembly and configuration.
13. Warranty and Support: Extended warranties and premium support services can increase the overall cost.

Steps to design a basic computer cluster using desktop PCs:

1. Choose Desktop PCs:

- Select desktop PCs that meet your performance requirements and budget constraints.

2. Select a Network Infrastructure:

- Decide on the network topology (e.g., Ethernet) and speed (e.g., 1 Gbps or 10 Gbps) for connecting the desktop PCs.

3. Install a Cluster Operating System:

- Install the cluster-ready OS like Linux distributions like CentOS, Ubuntu. on each desktop PC

4. Configure Network Services:

- Set up a static IP address for each desktop PC to ensure stable communication within the cluster.
- Configure DNS (Domain Name System) to enable easy access between cluster nodes.

5. Install Cluster Management Software:

- Choose cluster management software, to control task distribution and resource allocation among cluster nodes, for eg. OpenMPI.

7. Set Up Shared Storage:

- Configure shared storage so that all cluster nodes can access data and applications as needed, eg. NAS, SAN.

10. Monitor and Maintain (optional):

- Implement monitoring tools (e.g., Nagios, Zabbix) to keep an eye on the health and performance of cluster nodes.

Compute the global sum of 16 numbers using parallel distribution over 8 processors in a tree-structured manner, which involve both data-level and task-level parallelism.

Assuming you have 16 numbers: [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P] and 8 processors (P0 to P7).

Step 1: Data-Level Parallelism (Divide Data)

Split the input data into chunks, distributing them among the 8 processors:

P0: A, B

P1: C, D

...

P7: O, P

Step 2: Task-Level Parallelism (Local Sum)

Each processor computes the local sum of its assigned numbers in parallel:

P0: localsum = A + B

P1: localsum = C + D

...

P7: localsum = O + P

Step 3: Task-Level Parallelism (Reduce)

Perform a binary tree reduction to combine the local sums into intermediate sums:

Round 1: P0 + P1 → intermediatesum0

Round 2: P2 + P3 → intermediatesum1

...

Round 7: intermediatesum6 + intermediatesum7 → GlobalSum

At the end of this, GlobalSum will contain the sum of all 16 numbers.