Q1. What are the trends in HPC and parallel computer architectures?

Performance Trends in HPC:

The number of floating-point operations executed per second can be used as an indication for analyzing the performance trends of computers. Figure 1.1 below shows a steady increase in the number of floating-point operations executed/second.

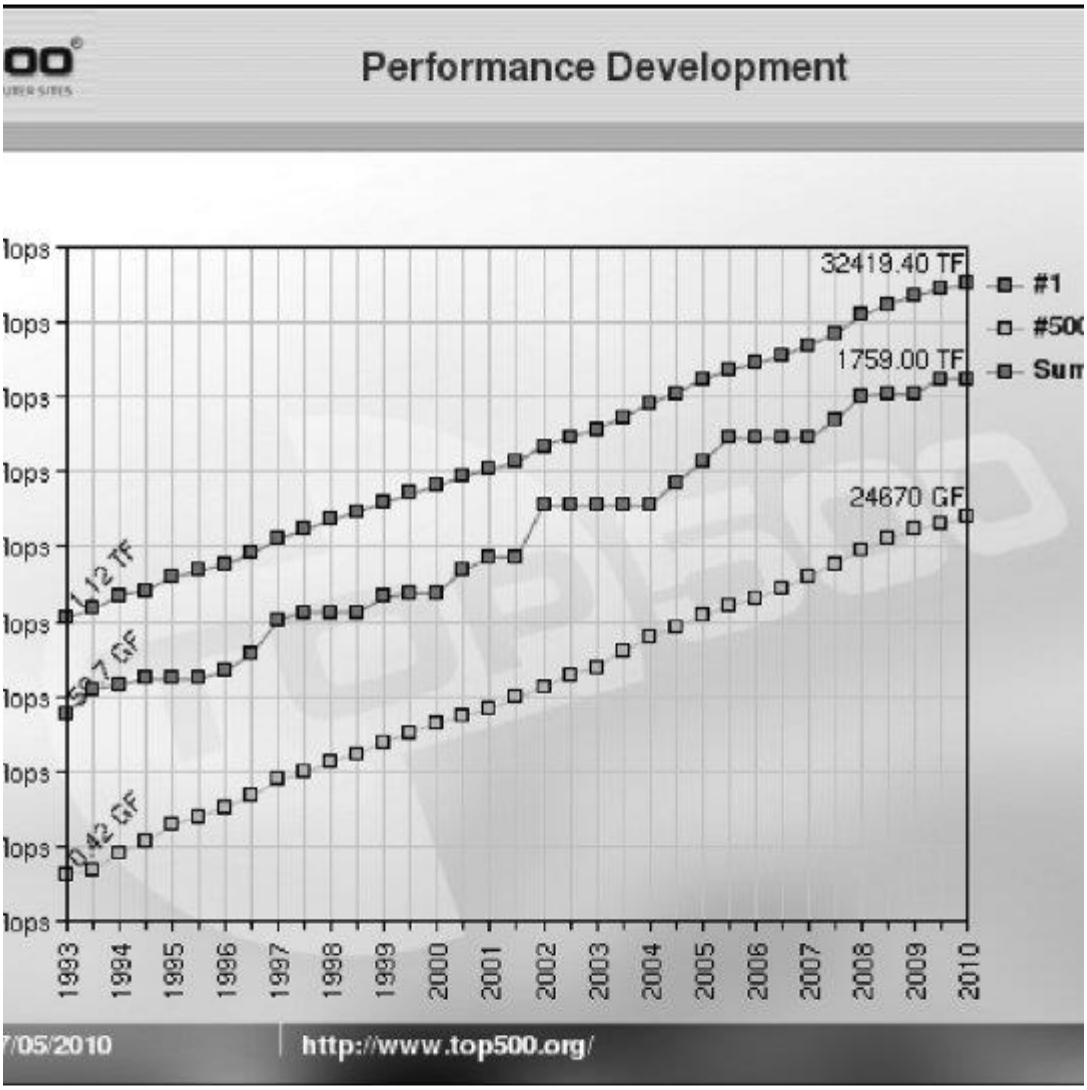
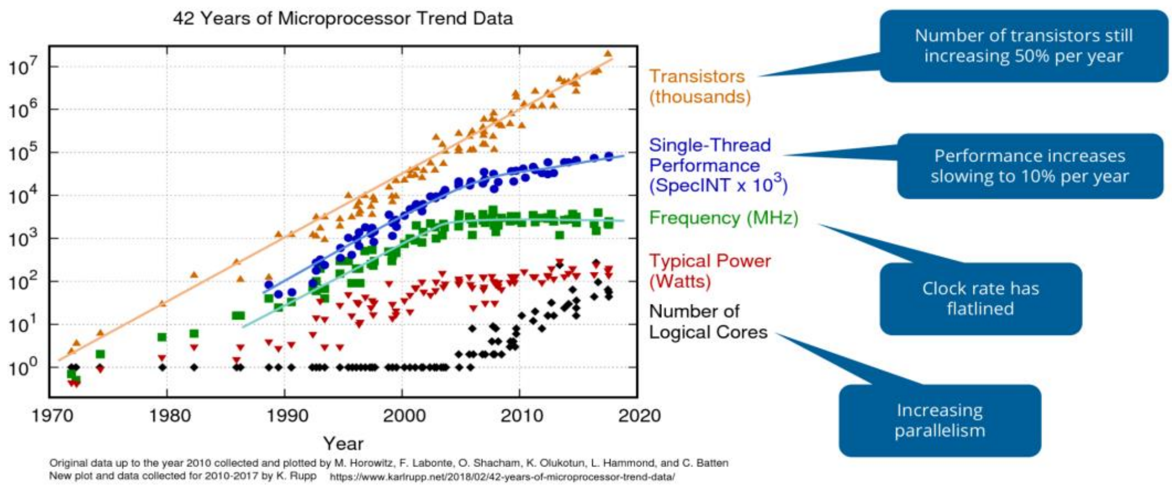


Figure 1.1

Technology trends in HPC:

According to Moore's law, the number of transistors that can be incorporated into a single chip will roughly double every 1.5 years (See yellow line in fig 1.2).

However, we notice that the same trend does not hold for other factors such as clock frequency and power consumption. In recent years, it has been observed that the clock frequency has flatlined due to power dissipation limits, so how is there still a performance gain? The answer lies in the shift to parallel computing, where the number of cores/processors has increased to compensate for the limitations to improving single-processor speeds. (See black dots, fig 1.2). By increasing the number of cores/processors, tasks can be distributed among the different cores/processors and executed in-parallel for better performance.



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

Figure 1.2

Shift to heterogenous architecture

Nowadays, it is popular to have a heterogenous architecture which includes both a CPU and GPUs (Graphics Processing Unit), for achieving high performance. GPUs have many more computing units when compared to CPUs, so can be used for achieving high computation throughput. However, each GPU core is not powerful as a single CPU, so applications generally use a combination of both CPUs and GPUs for the best performance.

Cluster computing

With the improvement of network connectivity, tasks can be distributed among multiple inter-connected computers working in parallel for performance improvement.

Q2. What is cache memory? What is the reason of introducing cache memories?

Cache memory is a type of fast, relatively small memory that is stored on computer hardware. The purpose of cache memory is to store program instructions and data that are frequently used by the computer during its general operations. Cache makes use of SRAM (faster), whereas main memory uses DRAM (slower).

Cache has generally smaller memory size when compared to the main memory because it is more expensive. It is also placed closer to the CPU compared to main memory and disk drives for faster access (See fig 2.1)

Cache is also usually present at multiple levels (Fig 1.3). Each level has a different memory size and access time.

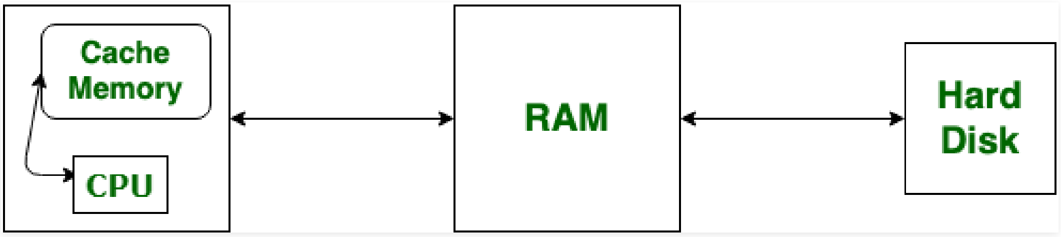


Figure 2.1

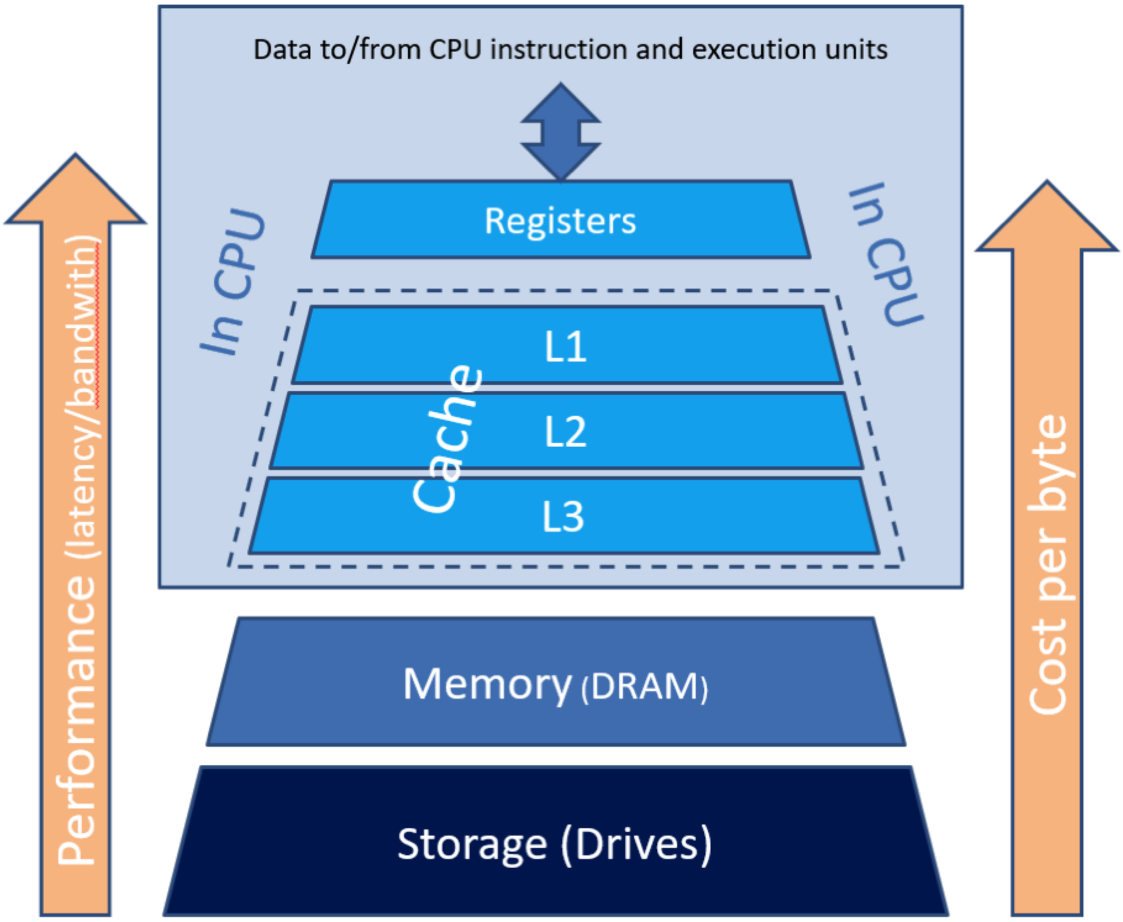

Data to/from CPU instruction and execution units
In CPU
Registers
In CPU
L1
Cache
L2
L3
Performance (latency/bandwith)
Memory (DRAM)
Storage (Drives)
Cost per byte

Figure 2.2

Advantages of using cache:

1. Fast access of frequently used data and instructions.
2. Can reduce the load on the data/instruction bus, allowing more processors to be connected to the bus.

Q3. What is the type of memory system/organization in large supercomputers?

There are two commonly used memory organization styles: shared and distributed memory. However, distributed memory organization is more commonly used in super computers.

In shared memory (fig 3.1), multiple processors have access to the same memory via an interconnection network. It has the following pros and cons:

Advantages of Shared Memory Programming

- Data sharing between processes is both rapid and uniform because of the proximity of memory to CPUs
- Insignificant process communication overhead
- Global address space offers a user-friendly programming perspective to memory
- No need to specify explicitly the communication of data between processes
- More intuitive and easier to learn

Disadvantages

- Difficult to manage data locality
- User is responsible for specifying synchronization e.g., locks
- Scalability is limited by the number of access pathways to memory
- Multiple processors compete for access to bus for memory access.

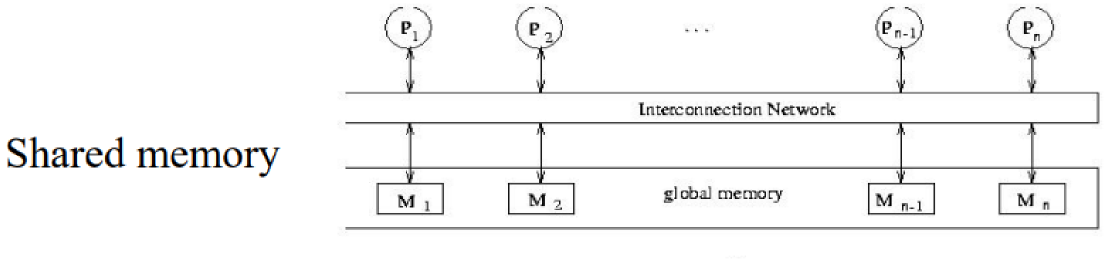


Figure 3.1

In the distributed-memory architecture, we take many multicore computers and connect them together using a network (fig 3.2)

With a sufficiently fast network, this approach can be extended to millions of CPU-cores and beyond. Shared-memory systems are difficult to build but easy to use, and are ideal for laptops and desktops.

Distributed-memory systems are easier to build but harder to use, comprising many shared-memory computers each with their own operating system and their own separate memory. However, this is the only feasible architecture for constructing a modern supercomputer.

Distributed memory has the following advantages and disadvantages:

Pros:

1. Easy to increase the number of processors/nodes
2. High data locality as each processor has its own local memory
3. Fast access to local memory

Cons:

1. Ensuring memory consistency among different processors is challenging as data needs to be sent to each processor's local memory separately.
2. Efficient use requires the interconnection network among different processors to be quite fast and reliable.
3. The design is more complex for managing compared to shared memory

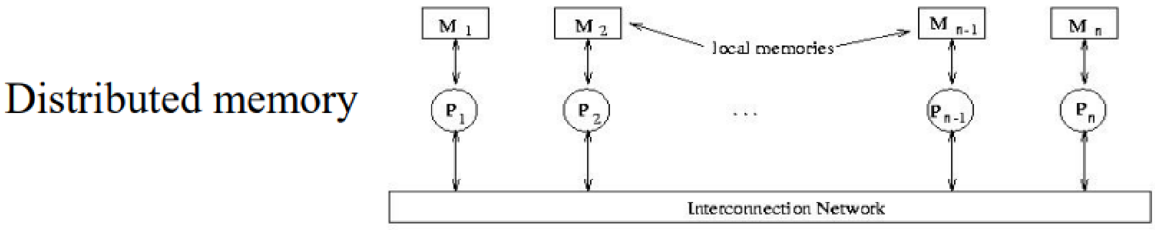


Figure 3.2