1 Key properties of HPC architecture

The content in this section is based on (Sterling et al., 2018, Chapter 2).

High performance computing architecture is the organization and functionality of its components and the logical ISA (instruction set architecture) it presents to computer programs that run on supercomputers or modest multiprocessors. HPC architecture exploits its enabling technologies to minimize time to solution, maximize throughput of operation, and serve the class of computations associated with large, usually numeric-intensive, applications. HPC has also been applied to big data problems, such as graph analytics. It also concerns reliability, cost, programmability.

- Three key properties of an HPC architecture that determine delivered performance.
  - The speed of the components comprising the system.
    - In this regard, a key parameter is the clock rate of its constituent processor cores, or basically the rate at which each executes instructions. Processor raw speed is often measured in clock speed, we use terms such as clock cycle, which is the time takes for a single clock tick; clock rate, which is usually measured in the number of clock cycles per second, i.e., Hz. For example, $1.5GHz = 1.5 \times 10^9$ clock rate.
    - The technologies employed for different functionalities, e.g. processor and memory; have differing speeds or cycle times. A big part of HPC architecture is about devising structures and methods that match up these different speeds, a good example is memory hierarchy consisting of a mix of slower higher-density DRAMs for capacity with faster low-density SRAMs called 'caches' to achieve speed.
    - The rate at which data can be transferred or communicated between any two points. Two measures are applied for this communication speed. 1) The bandwidth: determines how much information can be moved between two points in unit time, or the rate of data movement. 2) The latency: measures how long it takes to move data between the two points.
    - Bandwidth and latency can vary dramatically depending on the distances between the source and destination, as well as the type of technology employed and the amount used.
  - The parallelism or number of components that can operate concurrently doing many things simultaneously. In the case of architecture, we are looking at the number of independent components can operate independently, say the number of cores is a good measure. Parallelism is a key to performance because no matter how fast the speed of each independent part technology can be, it will never be fast enough on its own to deliver the necessary performance required by major application problems. For example, if a core is super fast, but a matching memory is slow, then your performance will be bound by the speed of the memory. Also, fundamental physical limits such as the speed of light, atomic granularity, (and the Boltzmann constant constrain) how fast a single processor core can execute a stream of instructions. Thus HPC architecture is heavily dependent on structures that permit many actions to occur simultaneously: the ability to do many things at once. This is referred to as "parallelism".
  - The third factor that determines delivered performance for a user workload is efficiency. Efficiency is primarily the utilization of the system, or the percentage of time that the critical components are employed. The efficiency is measured using utilization rate, that is, during the running of the system, what is the utilization rate? For example, if we are using these parallel components 90% of the time doing actual work or computation, then we can say, the efficiency is 90%.

$$e_{flops} = \frac{P_{sustained}}{P_{peak}} \tag{1}$$

where e_{flops} is floating point efficiency, and P_{peak} is the theoretical peak performance measured in flops, and $P_{sustained}$ is the achieved average floating point performance.

- **Power:** The speed of a processor is in part proportional to its clock rate, and which in turn relates to the power applied. Thermal control is essential if the system is not to overheat and ultimately fail as a consequence.
  - Air-cooling: Small to modest-size computers are air-cooled. (Cold air is forced through the system modules and over the processor, memory, and control sockets to remove the heat generated as they operate); Additional power is required to chill the air and force it through the system.
  - Metal radiator hardware: For higher wattage parts, substantial metal radiator hardware is fitted directly on to the sockets for thermal conductivity, providing greater surface area and cooling capacity.
  - Liquid-cooling: There is also a wide range of liquid cooling systems and mechanisms. For those higher-wattage parts, due to fitting metal radiator hardware, it reduces the packing density and hence the computing capability per unit module. Liquid-cooling can be used in this case to increase the packing density, and hence higher clock rates or larger die area dedicated to computing instead of cooling.
  - Active thermal control: Measurement of temperatures throughout the system or running of the system, including key chip temperatures, allows monitoring of thermal gradients and can support thermal control. Modern multicore processors permit variable clock rates (mention Turbo mode), voltage adjustment, and variable number of active cores, all of which facilitates achieving a balance between power and performance.
- **Reliability:** Errors can occur, no system operations are perfect.
  - Hardware fault: An error can be caused by hardware failures that are occurred in a core of a processor, or in the processor socket, it can also be in the memory, communication, or even harddrive, control unit etc.

    Hard fault: "Hard" faults occur when some part of the hardware breaks permanently; – If the error is from hardware error, the system has to be re-configured to eliminate the faulty hardware

    Soft fault: A "soft" fault occurs when a part intermittently fails but otherwise operates correctly; – If it is a soft error, the program can be restarted from the last checkpoint without reconfiguration.
  - Software fault: Of course, there is also software errors that are due to flawed coding in either user application, or system programs (such as OS); – the code will have to be corrected by the user or application supplier before it can proceed.
  - A common methodology on high-scale machines is "checkpoint/restart". That is periodically the system will stop a computation being performed and store all the program state at that point (checkpoint), usually on secondary storage. If an error occurs after a checkpoint the problem need not be restarted from the beginning but rather from the last checkpoint.
- **Programmability:** How difficult to write or develop a complex application code reflects the programmability of a system. Many factors contribute to programmability: including the processor core and system architecture, the programming models and facility of the language, the effectiveness of the system software such as compilers, runtime systems, and operating system, and the skills of the programmers.

  Many libraries of common codes have been developed by experts and optimized for a diversity of HPC system types and scales. Code reuse is critical to managing application development complexity

and difficulty. The discipline of "software engineering" provides principles and practices that guide overall control of workflow management, including testing. These and other methods contribute to programmability.

References

Sterling, T., Anderson, M., and Brodowicz, M. (2018). *High performance computing: Modern systems and practices*. Morgan Kaufmann.