용어설명

- task-level parallelism or process-level parallelism
  : utilizing multiple processors by running independent programs simultaneously.
- parallel processing program
  : A single program that runs on multiple processors simultaneously.
- multicore microprocessor.
  : a microprocessor containing multiple processors ("cores") in a single integrated circuit. Virtually all microprocessors today in desktops and servers are multicore.
- Shared memory multiprocessor (SMP)
  : a parallel processor with a single physical address space.

The difficulty of creating parallel processing programs.

- Software 가 힘들다.
  - 이유1 parallel processing program이 multicore 위에서 성능과 전력효율이 보장되지 않으면 의미가 없다.
  - 이유2 task가 각각 같은 양을 등분해야 하기 때문에. 만약 등분되지 않으면 어느 하나의 core는 idle 상태가 되기 때문이다.
  - 이유3 core 간의 통신이 너무 많아서 성능 저하 될 가능성 존재.

→ Scheduling, partitioning the work into parallel pieces, balancing the load evenly, time to synchronize, overhead for communication between parties ....

- Strong scaling
  : Speed up achieved on a multiprocessor without increasing the size of problem.
- week scaling
  : // while increasing the size of the problem proportionally to the increase # of processors.

- memory hierarchy can interfere with the conventional wisdom about weak scaling being easier than strong scaling
  - ex) if the weakly scaled dataset no longer fits in the last level cache of a multicore microprocessor, the resulting performance could be much worse than by using strong scaling.

SISD, MIMD, SIMD, SPMD, and Vector.

- SISD : Single Instruction stream, single data stream, single core
- MIMD : Multiple Instruction stream, Multiple Data streams. multi core.
- SPMD : Single Program, Multiple Data streams. The conventional MIMD programming model, where a single program run across all processors.
- SIMD : Single Instruction stream, Multiple Data streams. the same instruction is applied to many data streams as in a vector processor.
- original motivation behind SIMD. ① to amortize (상환하다) the cost of the control unit over dozens of execution unit.
  ② the reduced instruction handwiden and space.
  SIMD needs only one copy of the code that is being simultaneously executed, while message-passing MIMD my need a copy in every processors. and shared memory MIMD will need multiple instruction caches.
- SIMD는 for loop 내 array에서 장점↑ case switch에서 장점↓

⇒ data-level parallelism: Parallelism achieved by performing the same operation on independent data