High Performance Computing

Types of Parallism

Martin Raum

Granularity

There is a whole scale of granularity in parallel computing, reaching from fine to coarse.

Fine Small tasks in terms of code size and execution time. For instance, addition of the components of a vectors.

Intermediate For instance, calculation of a time-step for one region in a PDE solver.

Coarse Large tasks in terms of code size and execution time. For instance, processing one of a few hundreds of large measurement records.

Flynn's Taxonomy

Flynn's Taxonomy from 1966 distinguishes types of parallelism by what is common and distinct in a computation.

SISD Single Instruction Single Data.

SIMD Single Instruction Multiple Data.

MISD Multiple Instruction Single Data. (Rarely encountered).

MIMD Multiple Instruction Multiple Data.

This taxonomy is usual employed to describe hardware capabilities.

SISD: Single Instruction Single Data

No parallelism; every step processes one “unit” of data with one instruction.

This is the former days’ standard CPU.

SIMD: Single Instruction Multiple Data

A single instruction is applied to more than one "unit" of data. For example, pairs of floating point numbers that make up the components of a vectors:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \\ a_4 + b_4 \end{pmatrix}.$$

This type of parallelism is often synchronized in hardware, meaning that the results of the performed computation are available at the same time.

Vector instructions are one prominent example of such parallelism.

MIMD: Multiple Instruction Multiple Data

Independent, usually arbitrary, instructions are applied to more than one “unit” of data.

For instance computing simultaneously

$$a+b, \quad a+c, \quad \sqrt{d}.$$

Any kind of multi-node computing system exposes MIMD parallelism, but also cell processors and some of the modern Chinese accelerators.

SIMD and MIMD

Some modern architectures merge SIMD and MIMD parallelism. For example, all modern GPU-based accelerators, which consist of blocks of SIMD units that are connected in MIMD mode.

Processes and threads

From the perspective of the operating system, MIMD parallel software falls into two different categories.

Processes Isolated entities that have to communicate explicitly through interfaces provided by the operating system. A program can run as many processes, each instance corresponds to one process.

Thread Partially conjoined entities that in particular share heap memory, enabling implicit communication. A process can be split up into many threads.

What threads share

Threads share:

- pid (process id),
- code (the instruction section in memory),
- global variables and heap memory,
- file descriptors (accessed through file pointers).

Threads do not share:

- stack memory,
- “thread local” variables and memory,
- program counter.