TDT4200 - Problem Set 1

Einar Johan Trøan Sømåen

2013/08/28

1 Part I, Theory

a)

Explain why multi-core processors have become so popular the past few years, despite being harder to program.

We are reaching a limit as to how much faster we can go in terms of pure serial execution speed, additionally we are getting problems with "dark silicon", meaning that we can't really fire up an entire CPU for serial execution, without melting it, to a larger and larger degree, thus we have to reap the benefits that can be found from executing our code in parallel instead, to continue increasing the execution speed of our programs.

b)

Briefly describe the four kinds of parallel systems in Flynn's taxonomy (SISD, SIMD, MISD, MIMD), as well as SPMD.

Acronym	**Full Name**	**Explanation**
SISD	Single Instruction Single Data	Classical serial execution system.
SIMD	Single Instruction Multiple Data	Vector-processing, one instruction performed on a set of data (i.e. matrix-scalar multiplication can be done as a single instruction, where as many multiplications as possible of those involved then can happen in parallell) SSE/Altivec are examples of SIMD-additions to existing CPU-architectures.
MISD	Multiple Instruction Single Data	Rarely seen, unless you count the pipeline itself, in this case various execution units perform different operations on the same data.
MIMD	Multiple Instruction Multiple Data	Multiple execution units performing (potentially different) instructions on different parts of the data, the clue here is that on an instruction level the various execution units can be doing quite different things, to quite different sets of data
SPMD	Single Program Multiple Data	Is the application of splitting a problem into subproblems, and running these subproblems in paralell on various execution units, thus the data is split, but still controlled by the same program (consider calculating some numeric integral, the problem in one domain can be handled on some CPU, while some other domain is handled by some other CPU)

c)

Briefly describe the main difference between a shared-memory system and a distributed-memory system.

A shared memory system is for instance what your household variant computer usually can be viewed as if working with multiple threads today, all the memory is shared among the threads (and indeed also geographically placed in the same machine), a programming model for this type of system is OpenMP.

A distributed memory system on the other hand, has it's memory distributed geographically among the nodes, such that a single node might have some memory, but that is local to that exact node, and thus not shared with the other nodes, to gain access to this memory, you would need to communicate between the nodes (i.e. with MPI).