1 Parallel Programming Models (PM)

The PM has to talk about the language, because they have different levels of abstraction (Java vs. Assembly)

Quick Introduction

Process:	Program in execution processed by processor
	It can have 3 states: running, starting, waiting, terminated
	For a waiting process it is saved where he will continue execution
	Processor has Program counter to know which process' turn is next
	There is 1 per CPU
Thread:	To do things parallel in a process
	If 1 thread does IO-Block, the whole process is blocked, except if you use kernel threads
	There is one per Core

1.1 Programming in parallel

First start out with a sequential program, then adapt and search for parts which can be parallelized.

Control Model:	Do the same / different things in parallel
Data:	Can be shared / private
Synchronization:	Is generally done explicitly
Communication:	Is done via shared variables

1.2 Open Multi Processing (OpenMP)

Syntax: # pragma omp directive [clauses]

(e.g.	# pragma omp parallel private (var)	creates parallel part with a private variable
	# pragma omp for	creates parallel for loop
	# pragma omp section	does a single execution of following code
	# pragma omp for schedule (static,2)	everybody does 2 following itereations)

Mainly meant for task and data paralelism.

1.3 pThreads

Just normal threads, you can do anything.

1.4 High Performance Fortran (HPF

This is Fortran with added annotations.
Statements distribute data among nodes. Then something is programmed for everyone to do.
→only data parallelism is possible

- The Control is Data Driven
- The Synchronization happens implicitly

You can basically distribute an array over multiple nodes and run the same operation everywhere.
Different Arrays can be aligned with each other on different nodes. →both have same elements on same node.
It is not possible to run different tasks on different nodes. Everybody runs the same thing.

1.5 Message Passing Interface (MPI)

Consists of independent processes, that each have a number. They can pass messages between each other.
Everything is done explicitly. For example everybody computes for part of vector and then shares.

1.6 Threading Building Blocks (TBB)

Invented by MIT, bought by Intel.
Extension of C++, specify task and when/ how to run it. Don't specify amount of nodes.
Parallel actions defined in classes. Classes then run as parallel.

1.7 Cilk

Extension of C with simple statements. Mainly for Divide & Conquer.
Everything (including dynamic multithreading) through statements "spawn" and "sync"

1.8 Satin

Just like Cilk just based on Java.

1.9 Pattern Based Languages

Idea: A set of often used methods get implemented with templates →very high abstraction.

1.10 Map Reduce

Never forget this.

1.11 Comparing Programming Models

With certain criteria:

Programmability = Performance + Productivity + Portability

Portabilities:

Code Portability:	Can Code be moved to other machines
Parallelism Portability:	Does Parallelism work on other machines
Performance Portability:	Does Performance stay on other machines