Chapter 1

Patterns (def.): Valuable algorithmic structures that are commonly seen in efficient parallel programs.

Serialization (def.): The act of putting some sets of operations into a specific order. The benefits of serialization are:
- SIMPLICITY;
- DETERMINISTIC BEHAVIOUR.

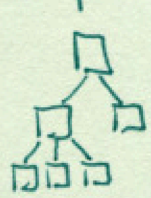
Serial trap (def.): Programming tools or constructs that make serial assumptions although they are not needed. A very common serial trap is the habit of evaluating an algorithm counting the RAM complexity, because a more likely bottleneck resides in the I/Os or communication.

Critical path (def.): It is the longest chain of tasks that need to be executed sequentially. The time needed for the execution of the critical path is called SPAN.

A good parallel algorithm has the shortest possible critical path.

Locality (def.): Memory accesses close in time and space are cheaper than those far apart.

Dependency tree (def.): Since some tasks need to be executed sequentially the set of tasks of an algorithm may be represented as a dependency tree.



Reduction (def.): It is the phase in which partial results are combined to form the final result.

Load balancing (def.): All workers have their fair amount of work to do

ISSUES OF THE 21ST CENTURY

Moore's law
The number of transistors that could be integrated on silicon chips double every 2 years

2005 3walls

Power wall
Increasing the clock rate lead to a too high energy consumption (either for working and for cooling)

Instruction level parallelism wall
Pipeline processors execute some "building-block" operations in parallel, but the amount of "building blocks" is not growing anymore. On the other hand, speculative compilation leads to more computation, hence more power consumption

Automatically extractable low-level parallelism has already been used-up

Memory wall
Although bandwidth may be improved the latency of memory operations is order of magnitudes bigger than operations executed on the processor

Vector parallelism (def.): A kind of data parallelism that takes place inside the processor and manages to execute one operation on multiple data at a time.

Parallel pattern (def.): Commonly recurring strategy to deal with particular problems.

Pointers are such a thing that makes parallelism a nightmare

Specificity ↓

DESIGN PATTERNS

ALGORITHM STRATEGY PATTERNS
OR
ALGORITHMIC SCHELETONS

IMPLEMENTATION PATTERNS

Semantics of pattern (def.): usage of such pattern to form a building block of an algorithm.
It goes without saying that different implementation choices MUST lead to the SAME semantics.

PATTERNS

- **Nesting**: Composition of more than one parallel pattern. Optional parallelism is MANDATORY.
- **Map**: Divides the problem in units that need only to be computed by one single function
- **Fork-Join**: Splits the problem in subproblems that can be solved in parallel and then recombines the pieces

DESIRED PROPERTIES OF PATTERNS (3 Ps)

- **Performance**
  - Achievable
  - Predictable
  - Scalable
  - Tunable
- **Productivity**
  - Expressive
  - Debuggable
  - Composable
  - Mantainable
- **Portability**
  - Functionality ← being able to run on a variety of devices
  - Performance ← same performance on different machines

Parallel Programming Models

- TBB – Intel Thread Building Blocks
- Cilk Plus
- OpenMP
- ArBB – Array Building Blocks
- OpenCL

Good Practices

- Use abstractions instead of mechanisms
- Try to achieve load balancing
- Use data parallelism, where the amount of work grows with the size of the problem
- Use STRUCTURES of ARRAYS instead of ARRAYS of STRUCTURES

Serially consistency (def.): it is the property of having the same behaviour of the sequential program.

Chapter 2

Data dependency (def.): one task cannot execute before some data it requires is generated by another task.

Control dependency (def.): certain events or side effects need to be ordered.

Data parallelism (def.): the bigger the dataset the more the tasks.
scales well

VS

scales poorly
Functional decomposition (def.): running different tasks in parallel.

Regular parallelism (def.): the tasks are similar and have predictable dependencies.

VS

e.g. a branch and bound algorithm
Irregular parallelism (def.): the tasks are dissimilar in a way that creates unpredictable dependencies.

Thread parallelism (def.): a mechanism to implement parallelism in hardware using a separate flow of control for each worker.

VS

Vector parallelism (def.): a single operation is replicated over a collection of data.
The elements of vector units are called LANES.
Vector may emulate threads using two different methods:
- MASKING: the vector execute both parts of the conditional statement and keeps only the "true" one.
  This approach may be optimized through coherent masking
- PACKING: first the condition is evaluated on all the FIBERS (data on lanes) and then they are reorganized in "true" and "false" cluster. At the end they are interleaved.

Task (def.): unit of potentially parallel work with a separate flow of control.

A task can have two different schedulings:

- PREEMPTIVE → no control on the execution timing;
- COOPERATIVE → a thread switches task only at a predictable switch point.

INSTRUCTION PARALLELISM

- Superscalar processor
  Executes multiple instructions in parallel as long as they do not depend on eachother
- Switch on event multithreading
  The hardware switches rapidly to a different hardware thread when a long-latency operation in encountered

When cores communicate they use caches thanks to the mechanism of cache coherence

MEMORY

- Hierarchy
  - Registers
  - Caches
  - RAM
  - Disk
- Bandwidth
  Amount of data transferred per unit of time
- Latency
  Amount of time needed to satisfy a transfer request
- Caches
  - Set-associative Compromise
  - Direct mapped
    a page has only one slot
  - Fully associative
    No structure

VIRTUAL MEMORY

Each process may use only a limited portion of memory (logical address space)

We want DATA LOCALITY for 2 reasons

- low fault rate
- rapid address translation
  [a large number of accesses in a short timeframe causes TLB thrashing]