parametric, reusable, portable
programming abstraction
modelling know, common, efficient
parallelism exploitation pattern

PARAMS → to distinguish different versions or usages → FUNCT / NON FUNCT

REUSABLE → ?

PORTABLE → across hw/sw archie
very different hw

VECTOR PROCESSORS
GPUs
HPC
COW/NOW
GRIDS

KNOWN, COMMON

EFFICIENT

SKELETON ADVANTAGE

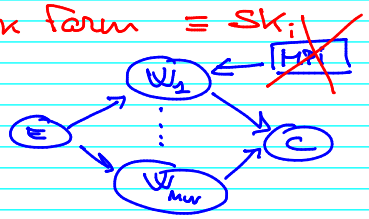
programmability (separation of concerns)
fast prototyping
correctness (termination)
↳ compositionality
portability — FUNCTIONAL
— NON FUNCT. (PERFORMANCE)

— debugging (fine tuning)
— optimisations
— performance models

WEAKNESS

lack of "configurability"
expandability
of the skeleton set

task farm ≡ SK_i



$$SK = \{ SK_1 \dots SK_m \}$$

? ?

DATA PARALLEL PATTERNS

1 complex data structure

↓ PARTITIONING

{ set of partitions }

$x_1 \dots x_n$

↓ f PAR

$\{f(x_1) \dots f(x_n)\}$

REBUILD

↓

UNIQUE RESULT

(COMPLEX DATA STRUCTURE)

REDUCE LATENCY

STREAM PAR. PATTERNS

SEQUENCE (in time)

$x_1 \dots x_n$ of data

↓ f PAR

$f(x_1) \dots f(x_n)$

STREAM (RESULT)

REDUCE SERVICE TIME

CONTROL PAR PATTERNS

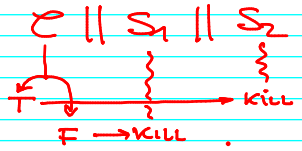
LOOPS

indep iterations

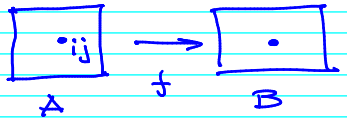
↳ PAR

IF c then S_1 else S_2

↳ SPECULATIVE PARALLELISM



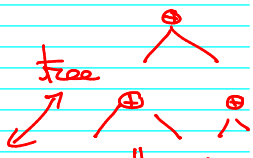
DATA PARALLEL PATTERNS

MAP 

$b_{ij} = f(a_{ij})$

REDUCE $\langle x_1 \dots x_n \rangle \xrightarrow{\oplus}$ $x_1 \oplus x_2 \oplus \dots \oplus x_n$ (⊕ commutative associative)

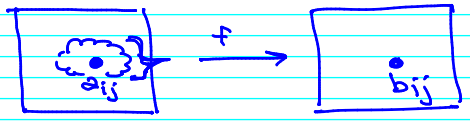
free



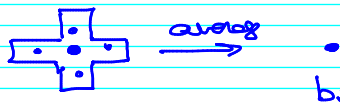
DIVIDE & CONQUER divide phase ($\rightarrow$ {subproblems})

conquer phase (sums up solutions)

STENCIL

$\forall i \forall j$ 

f: average $(a_{ij}, a_{i-1j}, a_{i+1j}, a_{ij+1}, a_{ij-1}) \rightarrow b_{ij}$



dimension of the stencil may be dynamic

SCAN $\langle x_1 \ldots x_m \rangle$ $\oplus$

$\downarrow$

$\langle x_1, x_1 \oplus x_2, x_1 \oplus x_2 \oplus x_3, \ldots, x_1 \oplus x_2 \oplus \ldots \oplus x_m \rangle$