# IMPLEMENTATION TEMPLATE

$Sk\,Set = \{Sk_1 \dots Sk_n\}$

$T_1^1 \quad T_1^2 \quad T_n^1 \dots T_n^{..}$

specific to architecture

$Sk_i \Leftrightarrow \begin{cases} T_i^1 \\ \vdots \\ T_i^t \end{cases}$

more than 1 templ ∃ ∀ target archie

we need at least 1 template x archie

TEMPL LIB :

< targ hw/sw archie , Sk , param process network , .... >

performance model

pipe | farm | seq

MC MPI

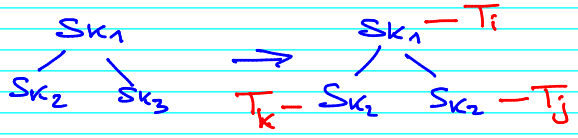
$P(p_1 \dots p_x)$

< MC MPI , {pipe farm}, * >

perf of the template when used with params $p_1 \dots p_x$

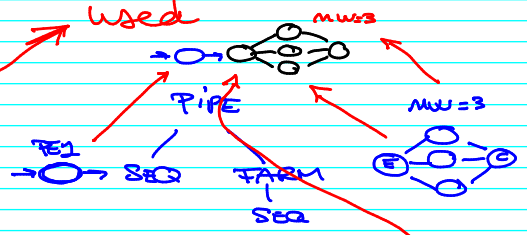
PHASE 1 PARSING → SK TREE (ABSTRACT SYNTAX)

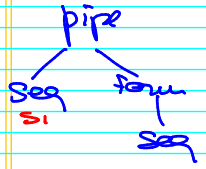
PHASE 2 TEMPLATE ASSIGNMENT → ANNOTATED SK TREE




Ski + target archie ⇒ set of templates that can be used

then we pick up the "best one"




P1 < pipe, ...
P2 < pipe, ...
F1 < farm, ...
F2 < farm, ...

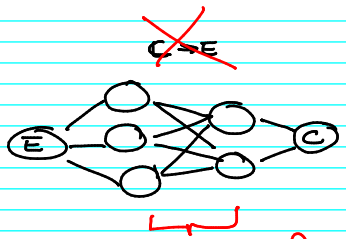
DURING THE "MERGE" I HAVE TO CONSIDER PERFORMANCE MODELS AGAIN

① PIPE
② SEQ (S1) PE1 will execute
③ FARM → F1 (solve the perf. model)
④ PIPE → P2

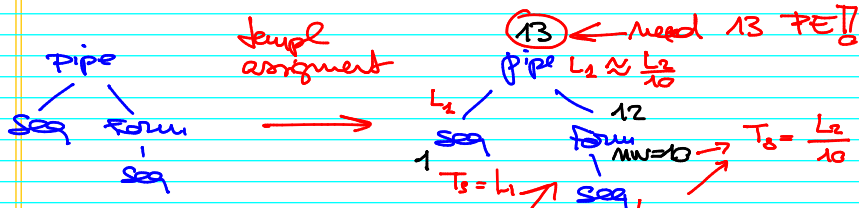
# PHASE 3 : OPTIMIZATIONS

PIPE(FARM, FARM)

OPT ?



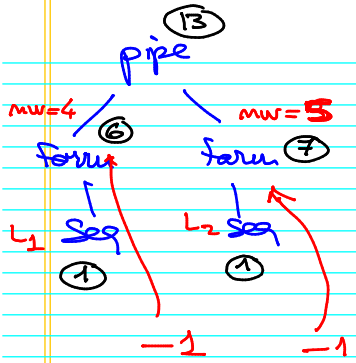

REDUCE PE REQUIREMENTS proper comm scheduling




program structure
Here is the place where I can take away res!

Hp : I have 10 PE

reduce while ensuring performance !




Hp = only 16 PEs

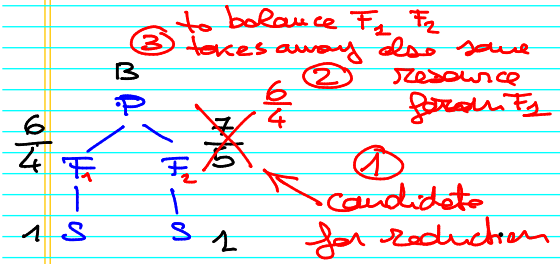
Loop: start from leaves

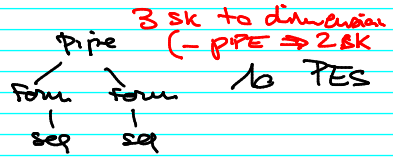
we take away
1 resource × node
(respecting perf. models)

compute the total PE
needed → TOT

if TOT ≤ available PEs
↳ OK DONE!

else loop again

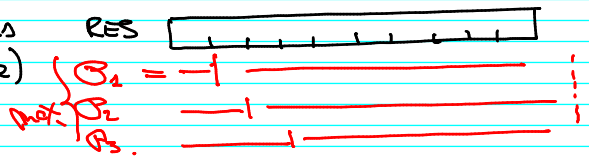


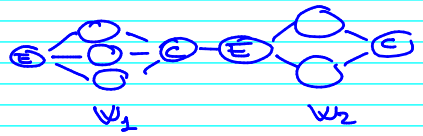

Alternative

free templates parameters RES (II degree)

n skeletons



# PHASE 4 : CODE GENERATION



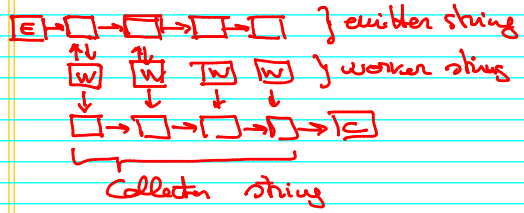

TARGET : POSIX/TCP COW/NOW

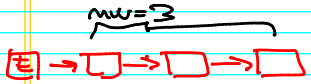
Same code ∀ → process — use comm lib for implementing communications

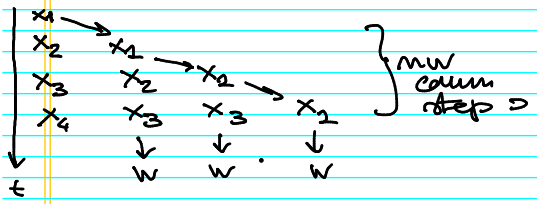
Same code { deployment, run

# TEMPLATE DESIGN

FARM TEMPLATE



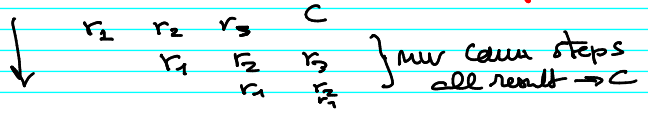




$T_w \approx nw\ T_e$ ← comm step time

when taxes are finished (on W)

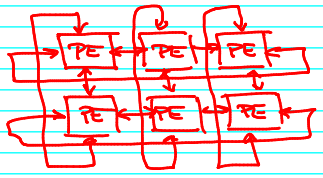
$W_i$ → results to the corresponding collector string element




TARGET ARCHITECTURE = MESH of PE

TILERA / PRO

8×8 matrix of cores on a single chip

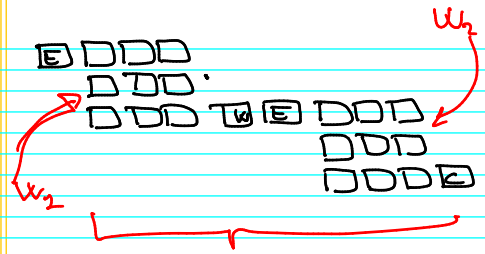



TRANSPUTER

T400 T800

POLARIS (INTEL)

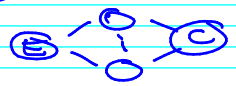
80 core EXASCALE 8×10 PROCESSOR




# TEMPLATE FOR MULTICORES

farm template

looks like the same as before




BUT:

1) → ○ → are threads (vs. processes)

2) comm are implemented through shared mem (vs. TCP/IP sockets)