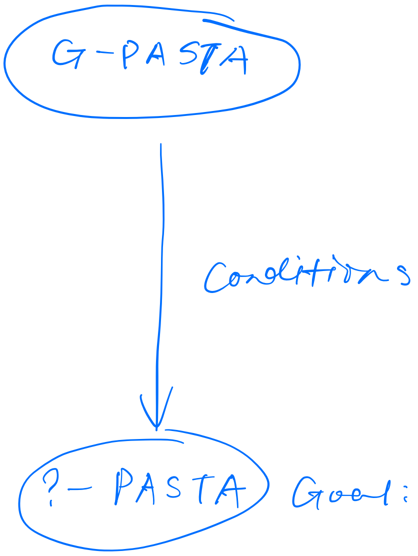
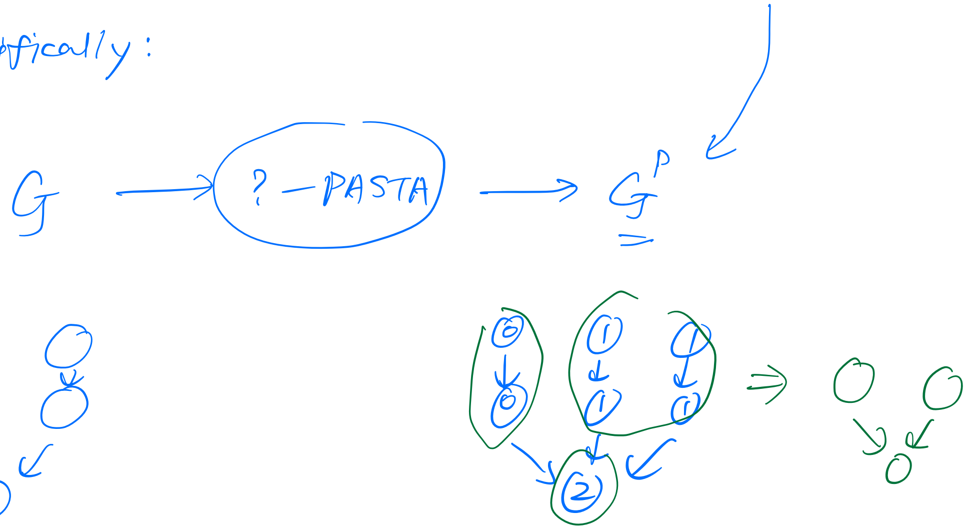
Current Situation:



Goal: constrained maximum parrallelism after partitioning

More specifically:

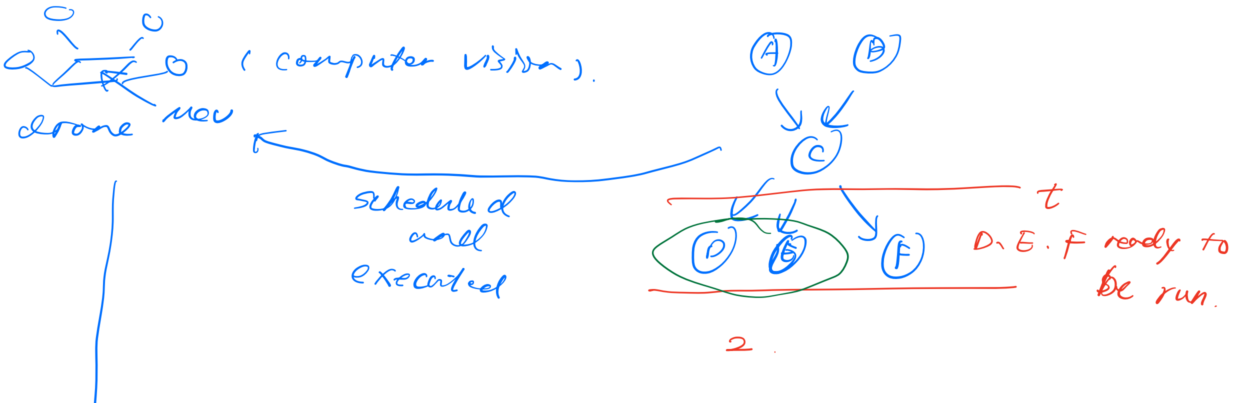


max par. = 3

constraint: max par. ≤ 2

Motivation: target on low-power environment
(or weak CPU or MCU)

example



↳

Assume MCU only has 2 threads ⇒

$$3 \text{ scheduling overhead } (T_S) . \quad 3T_S$$

$$+$$

$$\max(T_D, T_E) = T_D$$

$$+$$

$$T_F \quad = T_F$$

Total runtime if not partitioned

$$2 \, T_S$$

$$+$$

$$\max\, (\underbrace{T_D + T_E}_{\Delta}, T_F) \approx T_F$$

G-PASTA:

① Each source task will have a different par. ID (0 → N−1). N = # source tasks

② BFS, Each task will select the largest par ID from its dependents. (if this par. is full, start a new par. ID by ++ max_cur_par_ID).

↓

partition_size

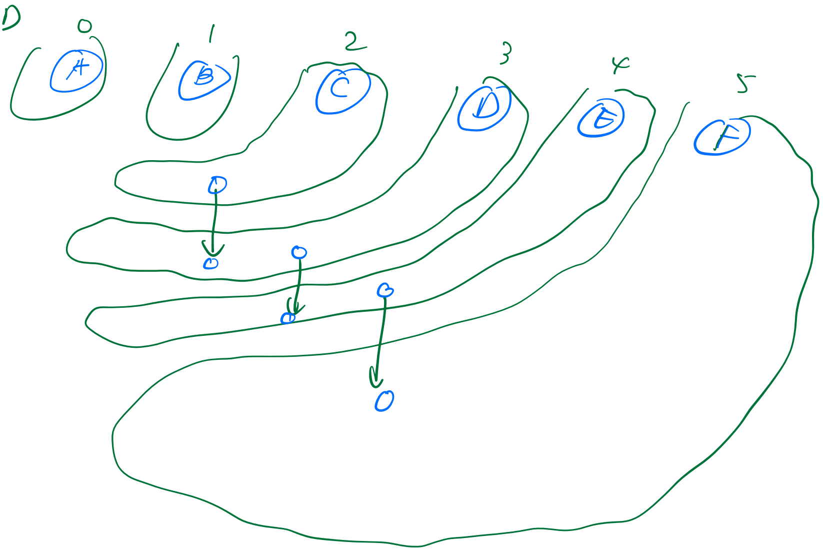
↓

which we set it as #tasks in the G.

Questions to verify:

☆ Is the max parallelism of G^P no larger than # source tasks in the G (If using G-PASTA).

Example: Par ID



partition_size = two

Way to verify. the question

① Set partition_size as #tasks in the graph. G = { node / edge }

② Use G-PASTA to partition G , get G^P = ⇑ { cnode / cedge. }

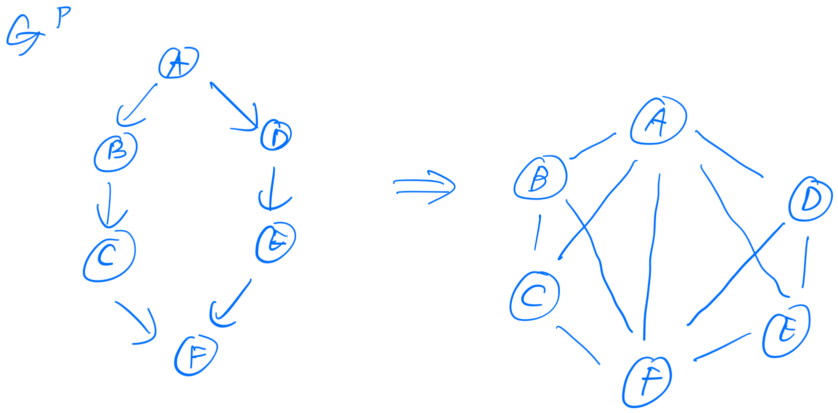
③ Check the max parallelism of G^P.

△ to see if it is less than #sources task in G.

(write a ctest, to check)

→ ref: check_cycle.

cmake(lists.txt)



(DAG)

undirected graph.

(B E)

(C D)

(C E)

(B D)

↙ ?

of

max parallelism = max size of (?) = 2