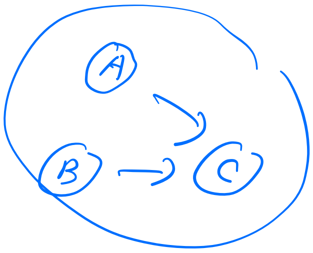
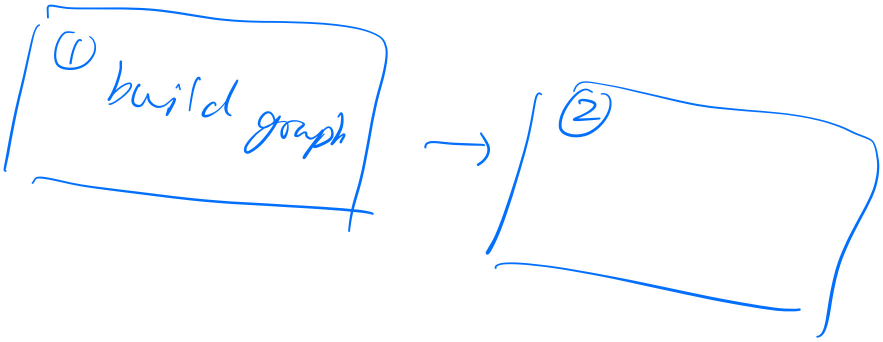
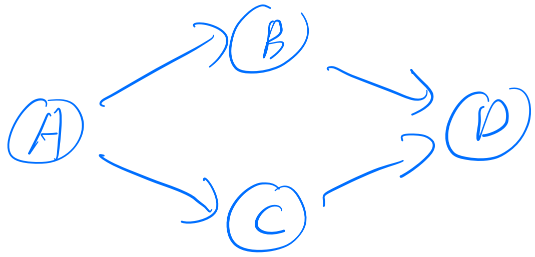
```
int a, b, c, d,e ;
c = a + b ; //A
d = a + c ; // B
e = c + d ; // C
```



scheduler

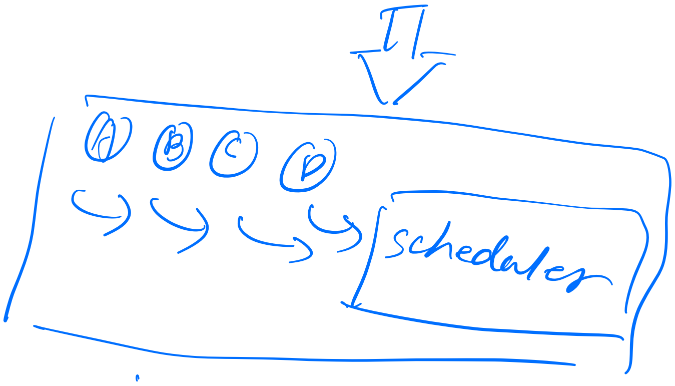


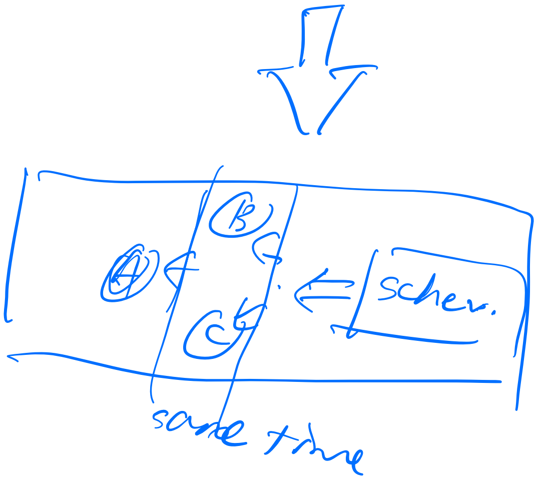
① Taskflow Workflow



Workflow

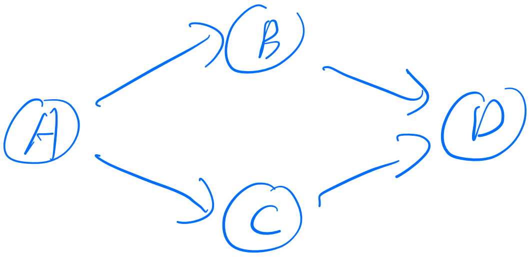
1. construct the graph → 2. execute the graph

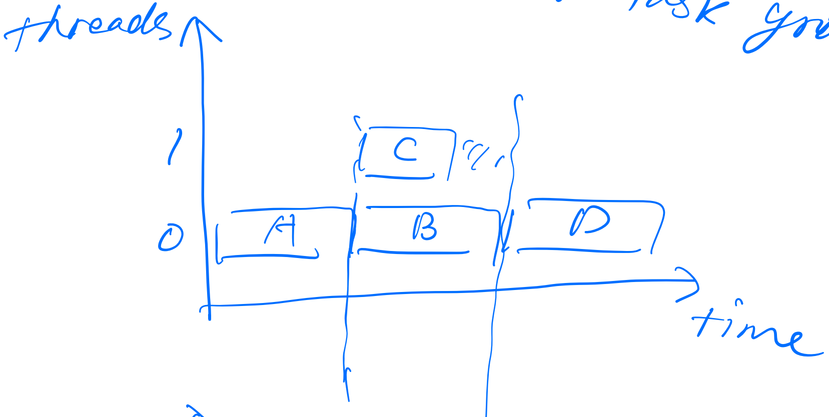


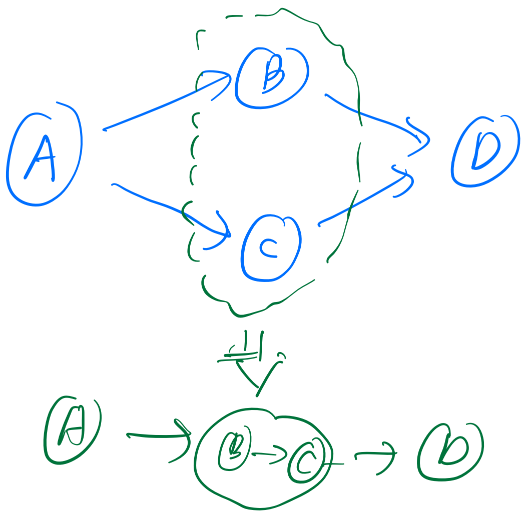


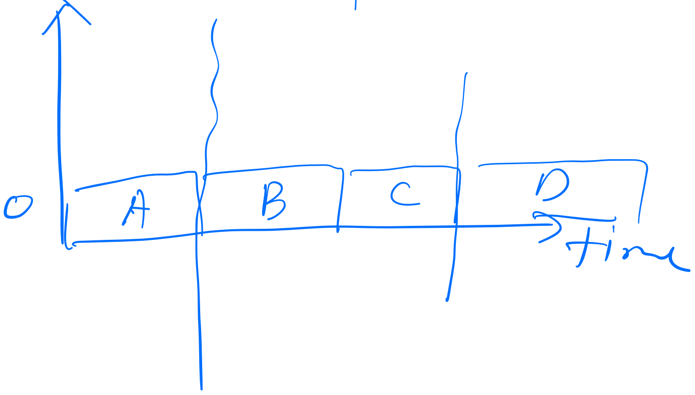
i.e. scheduling

② scheduling overhead. ∝ # tasks in ur task graph



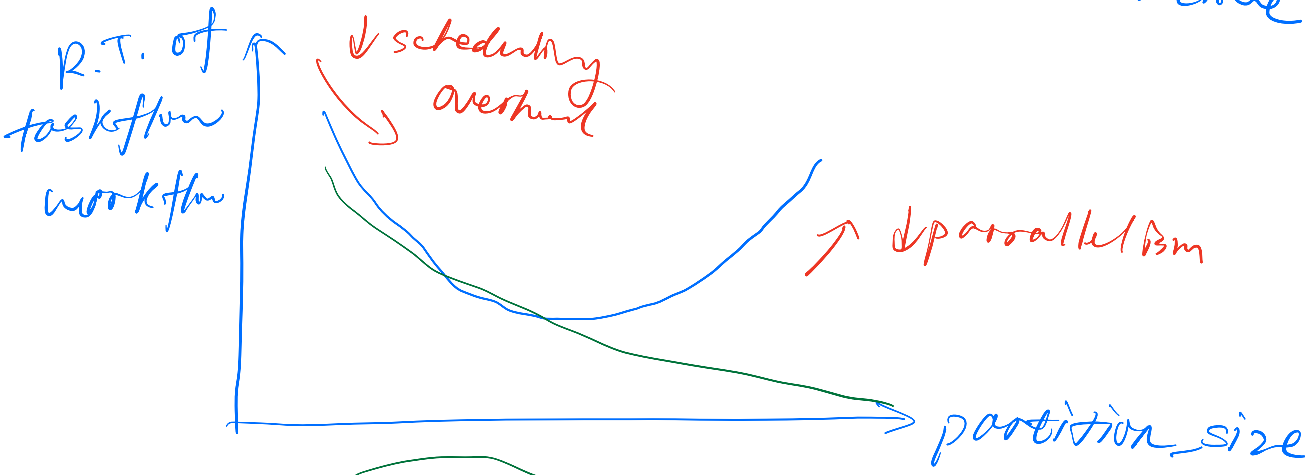


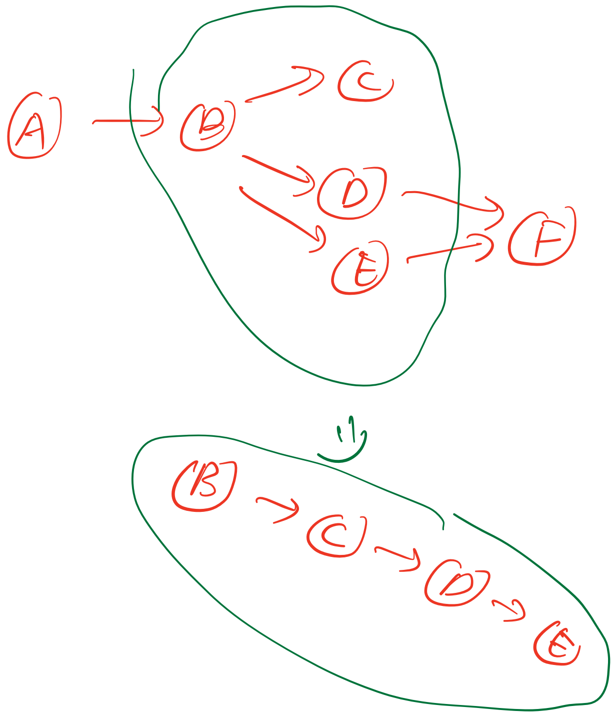




(3) trade-off : scheduling overhead vs parallelism

Assumption: Comparable scheduling overhead and task runtime





④ Task graph partitioning

task-graph based / task-parallel application

a. construct the graph → b. executing the graph

⇑

partition the graph

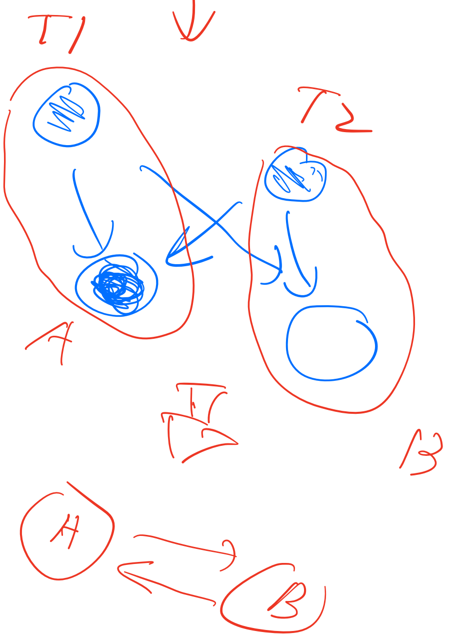
⇓

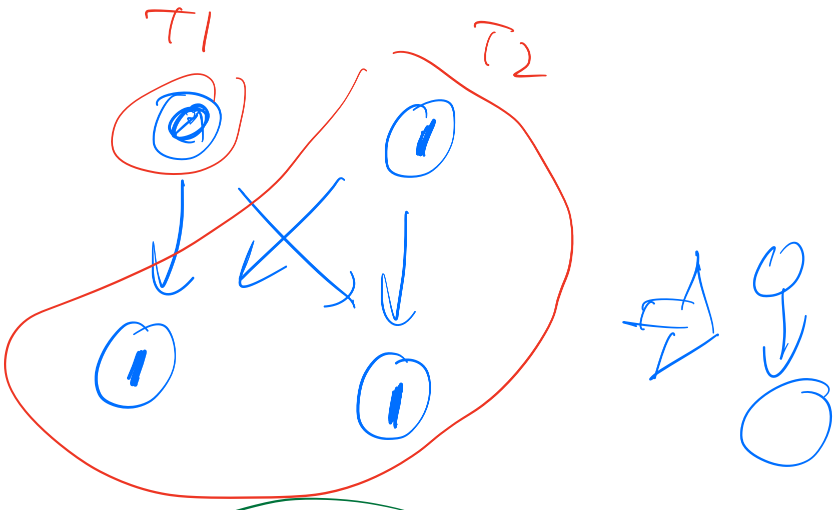
additional step needs to be fast!

⇓

do it in parallel

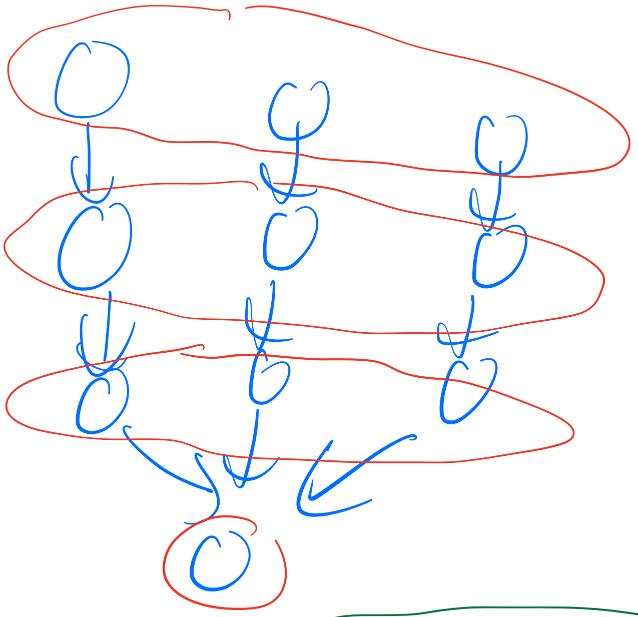
⇓

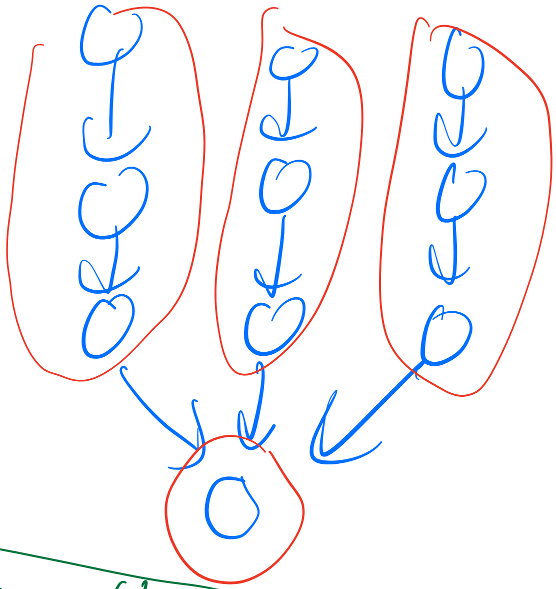




no cycle

a task will always choose the longest Par.ID from its dependents.





maintain parallelism

starts a different par ID

for each source node

Summary:

Partitioning Task Graph

① fast

② no cycle

③ maintain parrallelism