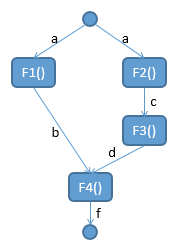
11. Patterns 9: Redegør for følgende concurrency mønstre

# Futures

En future er et resultat af en udregning som ikke kendes til start men bliver tilgængelig senere. Udregningen af en future kan ske parallelt. En future er en Task der returnere en værdi.

* Parallel task – asynkrone actions
* Futures – asynkrone funktioner.

De bruges når vi ønske at parallelisere kode med data afhængigheder.

* a = Some variable.
* b = F1(a)
* c = F2(a)
* d = F3(c)
* f = F4(b,d)
* f = Totalt output

Her kunne b udregnes som en future ved at køre F1 som en task: Task<string> future = Task.Run(()=> F1(a))

Task med en returværdi!

Og F4 kan så kaldes med: F4(future b.Result, d);

Derved kan F1 blive udregnet mens F2 og F3 udregnet.

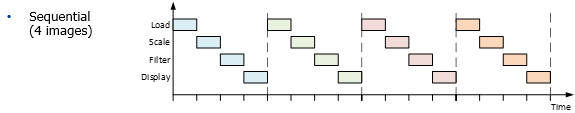
# Pipelines

En pipeline parallelisere proceseringen(databehandling) af en sekvens af input værdier. En pipeline består af en serie af producer/forbruger trin. Kan bruges når der er for mange afhængigheder til at lave parallelle loops.

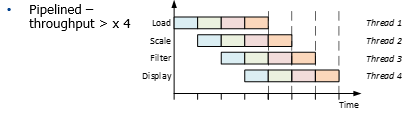
Proceserringen inddeles i paralleliserbare trin. Hvor output fra trin i er input til trin i+1. Ellers er trin uafhængige af hinanden.

For billede behandling kunne det være inddelt i fire trin:

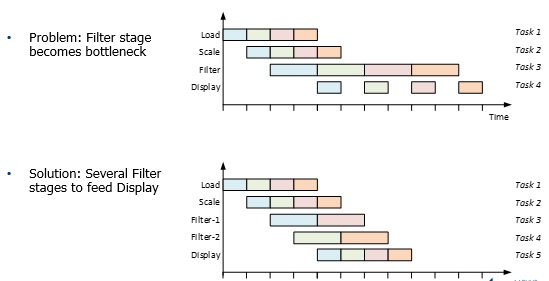
Her ses sekventiel behandling af billederne.



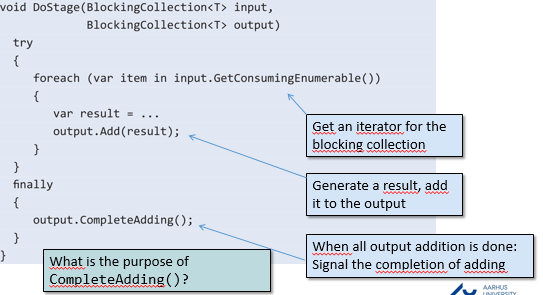
Her ses en pipelined udgave:



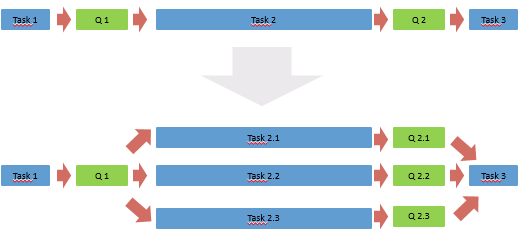
Hvad så hvis der kommer en flaskehals i pipelinen?



I c# laves det med tasks og concurrent queues (BlockingCollection<t>)



I C# vil det se sådan her ud hvis et trin har flere trin (tasks).



Her er det så forbrugeren der er ansvarlig for at tage ud af alle køerne.

