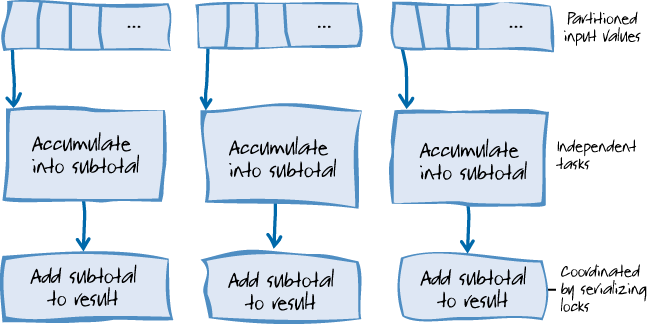
10. Patterns 8: Redegør for følgende concurrency mønstre

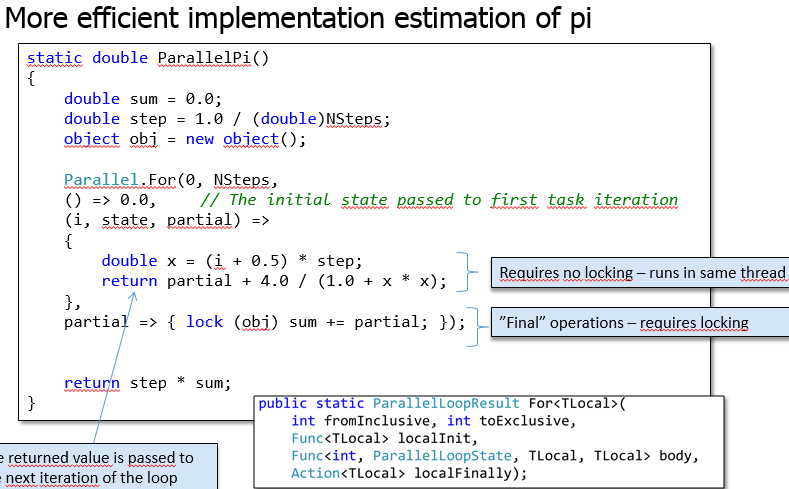
# Parallel Aggregation – AFHÆGIGE ITTERATIONER

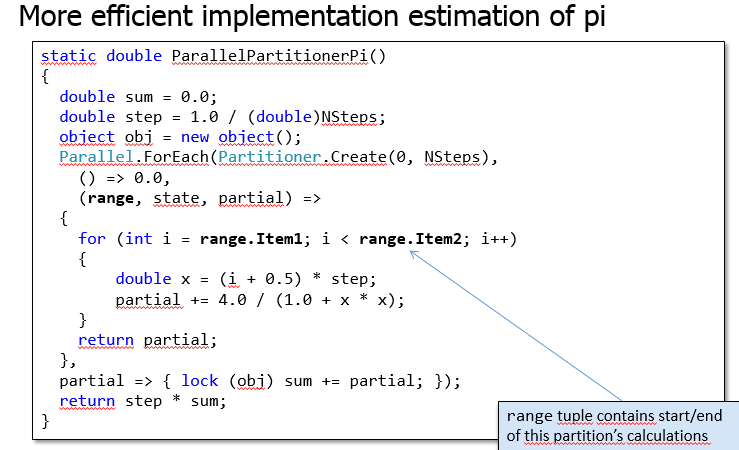
Når der arbejdes parallelt er der ofte behov for at opsamle data, og når det skal ske på tværs af tråde så er vi nød til at synkronisere den variable som vi opsamler data på.

Det kunne være et loop der udregner en sum dette har ikke uafhængige steps. Men der er stadig en måde at løse det på via parallel Aggregation pattern.



Det handler om at dele inputtet op i klumper og derefter udregne sub total for hver klump for til sidste ende at lægge summen sammen.





# MapReduce

Map Reduce er et pattern der tillader parallelle udregninger på store data sæt.

Der er 4 steps i Map Reduce

* 1. Distribute – Deler data ud til de forskellige noder der skal lave beregninger for at kunne arbejde parallelt (Forskellige PCer)
* 2. Map – Laver key value pairs på data sættet.
* 3. Group – Samler key value pairs så alle values der har samme key bliver samlet i et key value pair hvor value har alle de value der var i de første key value pairs.
* 4. Reduce – Aggregere data til det svar vi ønskede fra den oprindelige forespørgsel.

Som eksempel kunne man se på energi produktionen for sol celle anlæg her kunne vi spørge hvor meget EL er der blevet produceret for hver time i døgnet.

1. Distribute – vil her været at dele logfilerne ud til de forskellige noder
2. Map – Kunne være at tage alle entries for energi ved timen 12 også lave det til et key value pair.
3. Group – Her samles alle key values så key 12 vil indeholde alle de aflæste kW
4. Reduce – Her ville vi så summere alle kW for f.eks. Key 12.

