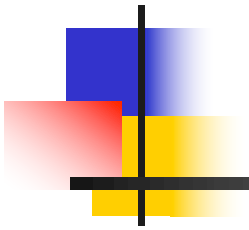


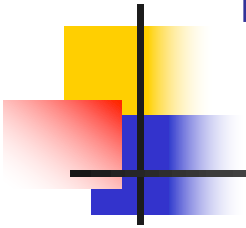
IN3030/IN4330
Effektiv parallellprogrammering
L02, (Uke 4) våren 2024

Eric Jul
Professor
Gruppeleder Programmeringsteknologi
Institutt for Informatikk
Universitetet i Oslo
Norge

Resume of first lecture v2024

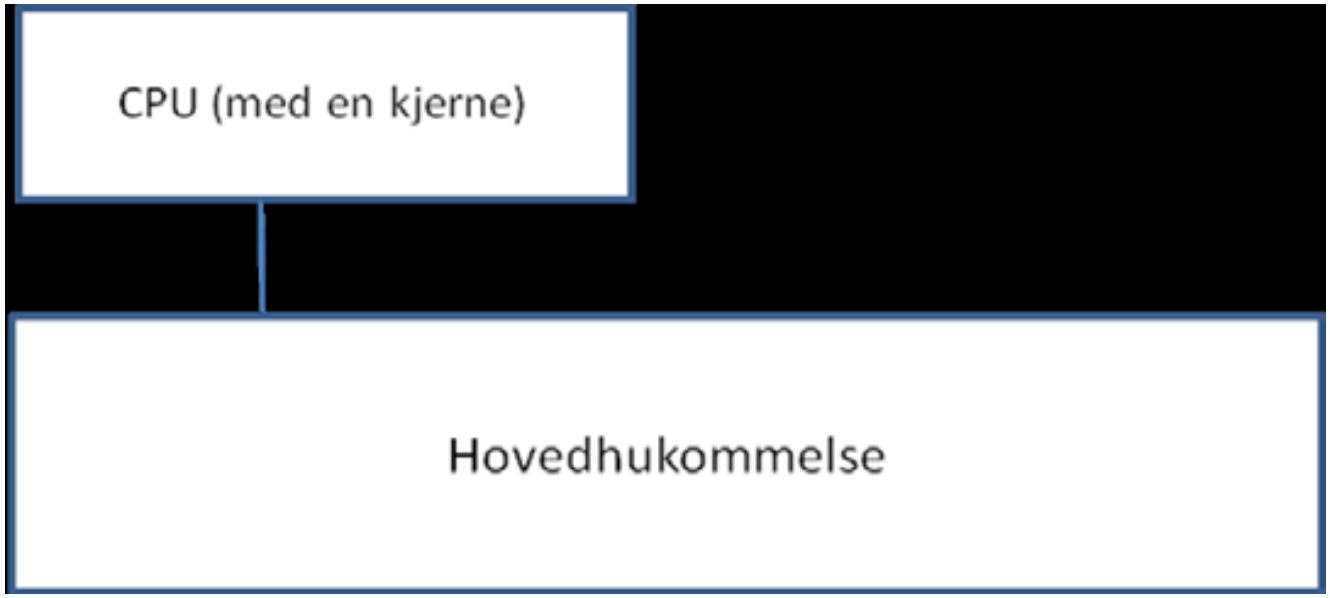
- Motivation
  - Utilization of Multi-core – by changing sequential programs into parallel programs
  - Multi-core hardware driving this trend
- Purpose
  - Convert sequential program into parallel versions
  - Achieve speedup
- Requirements
  - Correctness – Effective!
  - Efficient – MUST be FASTER – measured by speedup
- Approach
  - Empirical – *the proof of the pudding is in the eating!*
  - *Timings have the ULTIMATIVE SAY!*
  - *(Theory is fine; but practise is essential!)*

Resume of first lecture v2021 (second page)



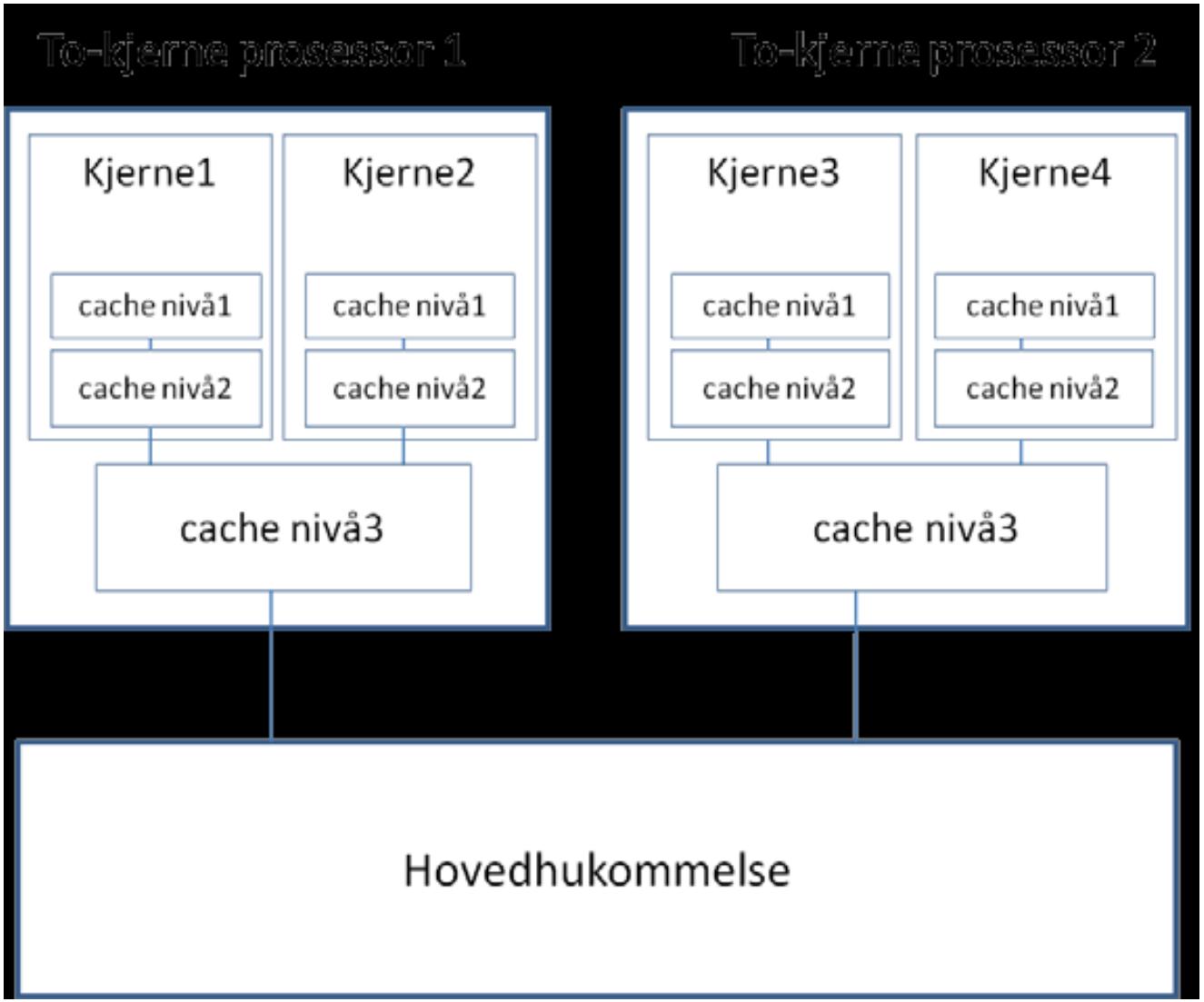
- Central metric: SPEEDUP
  - Speedup: sequential time / parallel time
  - Want speedup > 1
  - Really want speedup = *number of cores*
- How?
  - Parallel threads in Java
  - Must synchronize
- Evaluation?
  - Real-time clock times!!
- Multi-core architecture
- Non-uniform memory access
  - Multi-level caching
- Threads in Java

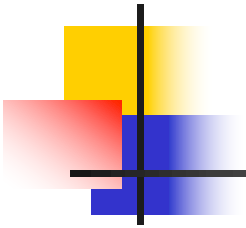
Maskin 1980 (uten cache)



Figur 19.1 Skisse av en datamaskin i ca. 1980 hvor det bare var én beregningsenhet, en CPU, som leste sine instruksjoner og både skrev og leste data (variable) direkte i hovedhukommelsen. Intel 8080: 1 MHz CPU 64 kbyte minne

Maskin ca. 2010 med to dobbeltkjerne CPU-er

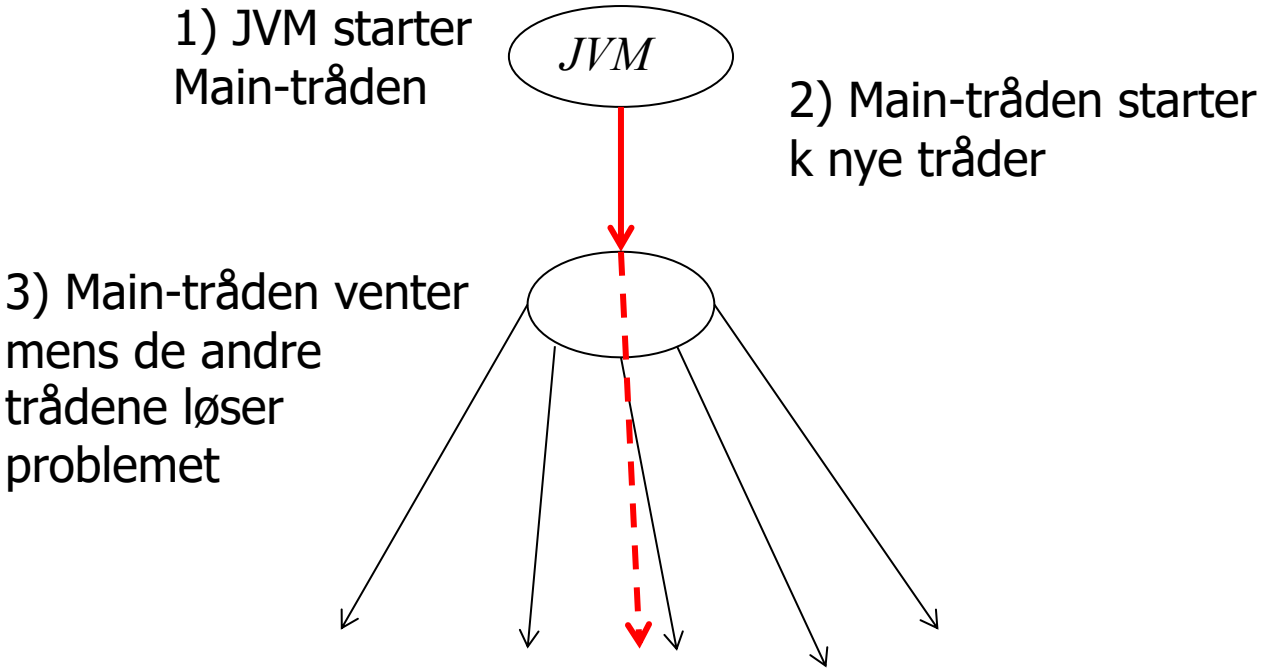




Hva er tråder i Java ?

- I alle programmer kjører minst en tråd – main tråden (starter og kjører i public static void main).
- Main-tråden kan starte en eller flere andre, nye tråder.
- Enhver tråd som er startet, kan stoppes midlertidig eller permanent av:
  - Av seg selv ved kall på synkroniseringsobjekter hvor den må vente
  - Den er ferdig med sin kode (i metoden run), terminerer da
- Main-tråden og de nye trådene går i parallell ved at:
  - De kjører enten på hver sin kjerne
  - Hvis vi har flere tråder enn kjerner, vil klokka i maskinen sørge for at trådene av og til avbrytes og en annen tråd får kjøretid på kjernen. «**Time Slicing**»
- Vi bruker tråder til å parallellisere programmene våre

>java (også kalt JVM) starter main-tråden som igjen starter nye tråder



Tråder i Java er objekter av klassen Thread.

Tråder i Java

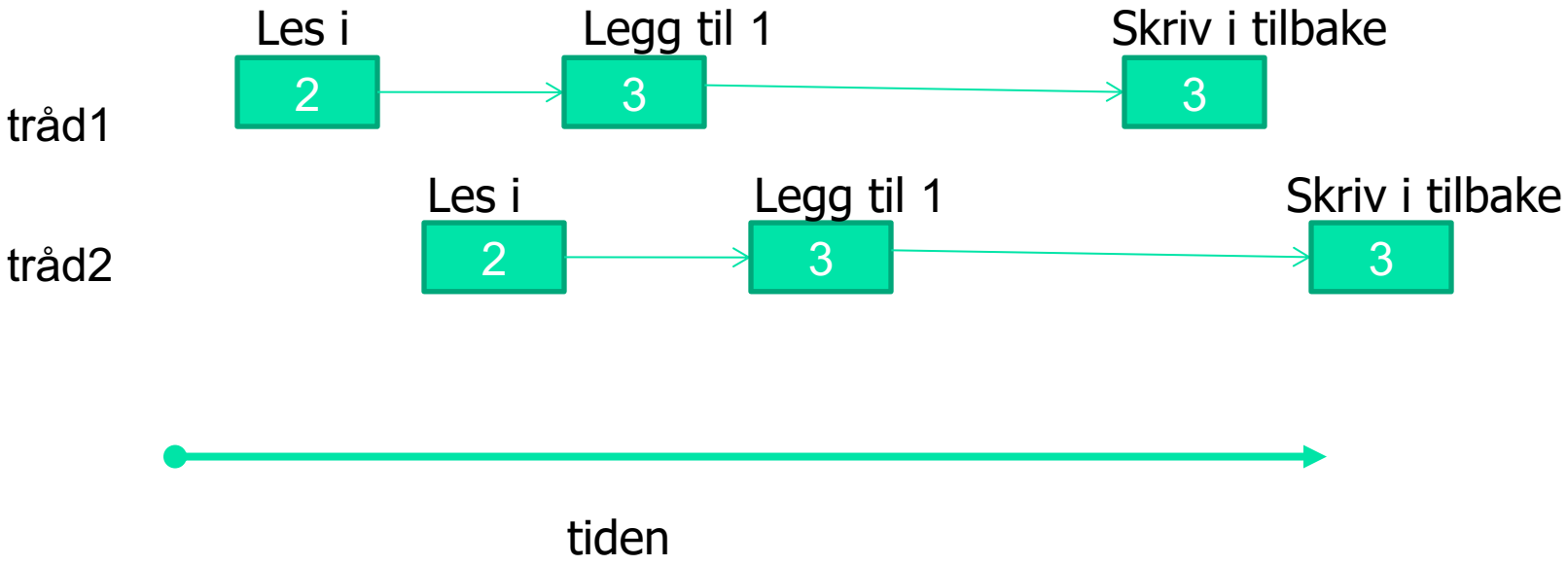
- Er én programflyt, dvs. en serie med instruksjoner som oppfører seg som ett vanlig, sekvensielt program – og kjører på én kjerne
- Det kan godt være (langt) flere tråder enn det er kjerner.
- En tråd er ofte implementert i form av en indre klasse i den klassen som løser problemet vårt (da får trådene greit aksess til **felles data**):

```
import java.util.concurrent.*;
class Problem { int [] fellesData ; // dette er felles, delte data for alle trådene
        public static void main(String [] args) {
                Problem p = new Problem();
                        p.utfoer();
                }
                void utfoer () { Thread  t = new Thread(new Arbeider());
                    t.start();
                 }

            class Arbeider implements Runnable {
                int i,lokalData; // dette er lokale data for hver tråd
                    public void run() {
                            // denne kalles når tråden er startet
                    }
             } // end indre klasse Arbeider
} // end class Problem
```

1) Ett problem i dag: operasjoner blandes ved samtidige oppdateringer

- Samtidig oppdatering - flere tråder sier gjentatte ganger: **i++ ;** der i er en felles int.
  - **i++** er 3 operasjoner: a) les i, b) legg til 1, c) skriv i tilbake
  - Anta i =2, og to tråder gjør i++
  - Vi kan få svaret 3 eller 4 (skulle fått 4!)
  - Dette skjer i praksis !



Test på i++; parallell

- Setter i gang **n tråder** (på en 2-kjerner CPU) som alle prøver å øke med 1 en felles variabel int i; 100 000 ganger uten synkronisering;

```
for (int j =0; j< 100000; j++) {
        i++;
}
```

- Vi fikk følgende feil - antall og %, (manglende verdier).
  Merk: Resultatene *varierer også mye* mellom hver kjøring :

Antall tråder n		**1**	**2**	**20**	**200**	**2000**
Svar	1.gang	100 000	200000	1290279	16940111	170127199
	2.gang	100 000	159234	1706068	16459210	164954894
Tap	1.gang	0 %	0%	35,5%	15,3%	14,9%
	2. gang	0%	20,4%	14,6%	17,7%	17,5%