Computer Arkitektur og Operativ Systemer

Denne forelæsning optages og gøres efterfølgende tilgængelig på Moodle
MEDDEL VENLIGST UNDERVISEREN, HVIS DU IKKE ØNSKER, AT OPTAGELSE FINDER STED

This lecture will be recorded and afterwards be made available on Moodle
PLEASE INFORM THE LECTURER IF YOU DO NOT WANT RECORDING TO TAKE PLACE

Computer Arkitektur og Operativ Systemer

Concurrency 1:

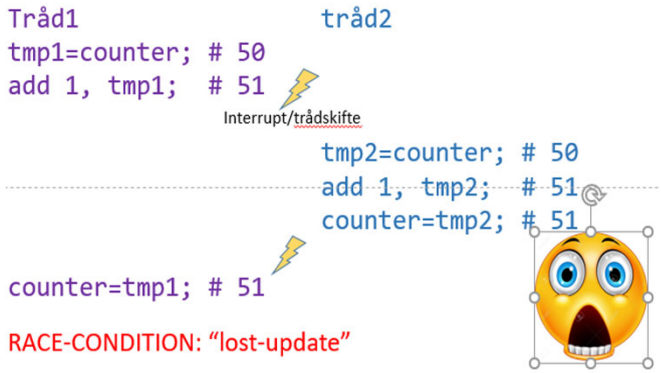
tråde, race-conditions, mutexes

Forelæsning 11
Brian Nielsen

Credits to
Randy Bryant & Dave O'Hallaron (CMU)
Youjip Won (KAIST)

Mål

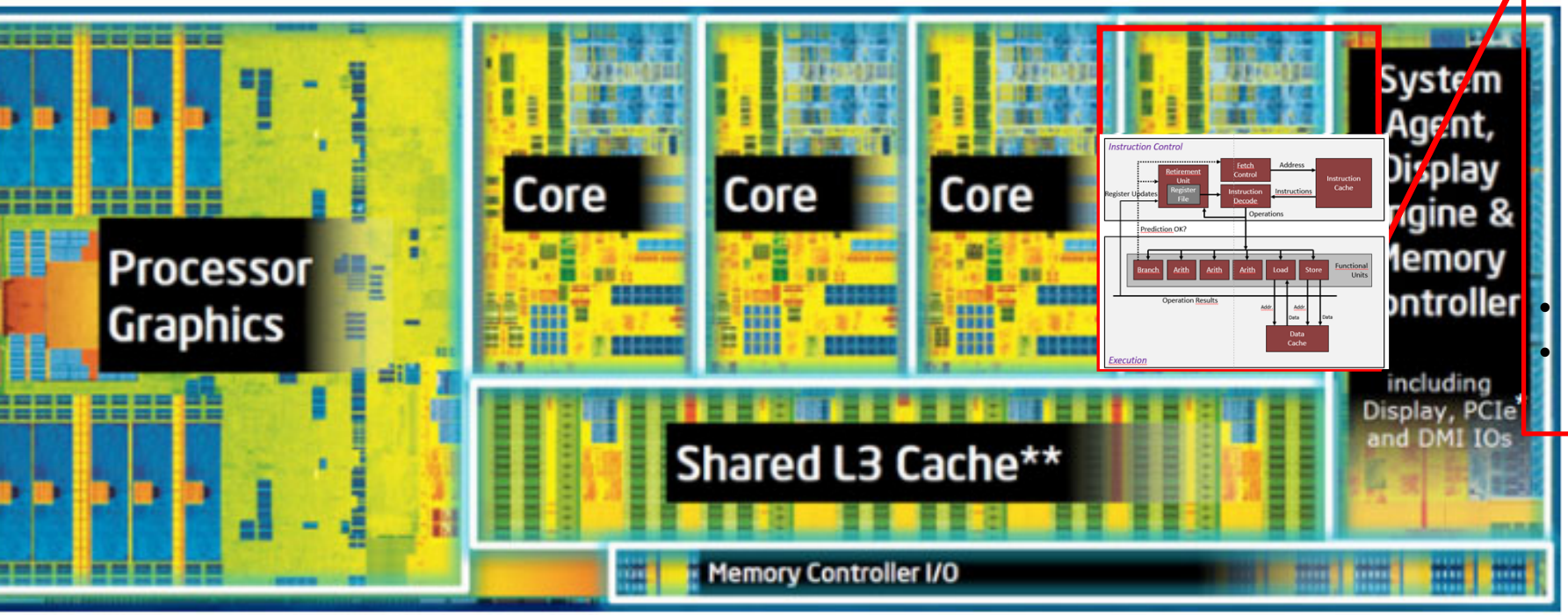
- Hvad er tråde? Hvad er de nyttigt til?
- Realisering: tråd kontrol blok, tråd-tilstande
- Vigtige begreber omkring samtidighed/ ***concurrency*** / sideordenet afvikling
  - Uforudsigelig relativ afviklingshastighed og rækkefølge (Non-determinisme)
  - Ubestemt (indeterminate) program udfald
  - Race-conditions
  - Kritiske regioner
  - Gensidig-udelukkelse, mutex, atomiske operationer
- Realisering af en ”lock” mekanisme
- Bruge af locks til sikring af gensidig udelukkelse i simple samtidige data-strukturer
- Anvende et API her (pthreads API) til programmering af concurrency




```
1    lock_t mutex;
2    . . .
3    lock(&mutex);
4    balance = balance + 1;
5    unlock(&mutex);
```

Multi-core Parallelitet

- Core-i7: op til 8 kerner (i9:18, Xeon 28, AMD EPYC: 32)



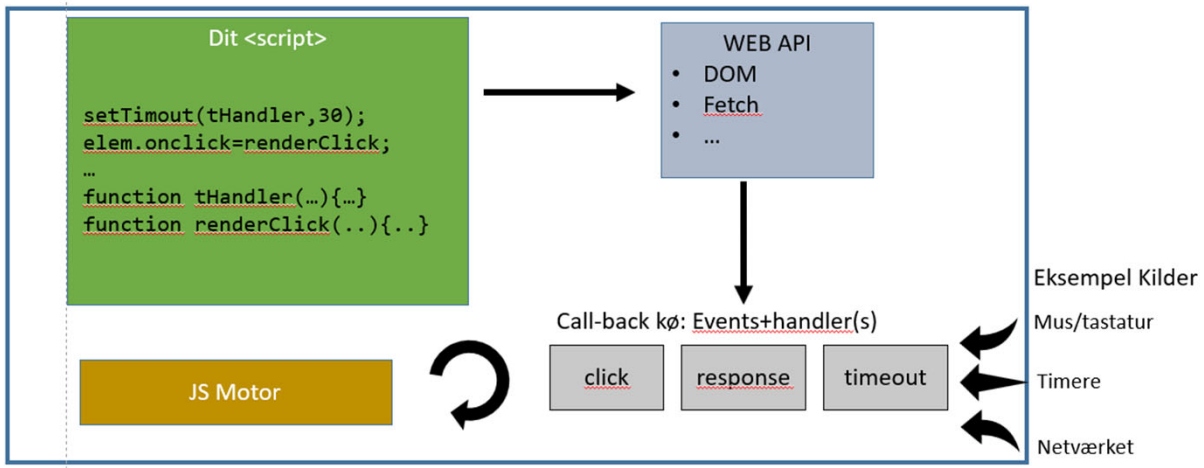
CPU chip indeholder flere processor kerner

- Egen register-bank
- Egen kontrol enhed
- Egne (pipelinede) funktionelle enheder (fx haswell i7):
  - 4 long add
  - 1 long mult
  - 1 long div
  - 1+2+1 FP
  - 2 Load +1 store
- Egen L1+L2 cache
- Andre ressourcer er delte: I/O, grafik,L3, RAM

- Instruktionsniveau-parallelitet er implicit, fra samme sekventielle “tråd”
- Udnyttelse af multiple-cores til parallelitet i programmer
  - oftest explicit tråd-programmering,
  - parallelliserende compiler: `for (x=1 .. N) in PARALLEL DO {}`

Side-bemærkning: Concurrency i Javascript

- Event-baseret asynkron afvikling: rækkefølge af events bestemt udefra
  - Hver handler-funktion afvikles udelt til ende ”run-to-completion” semantik

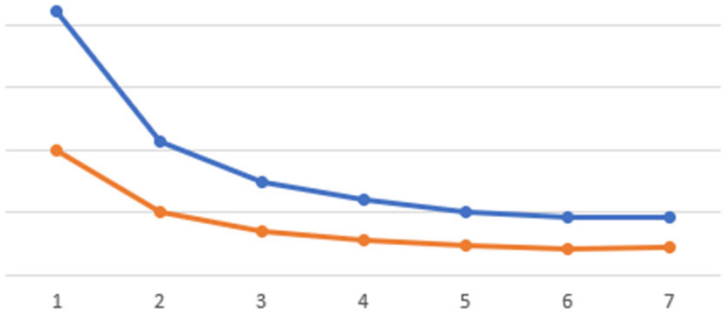


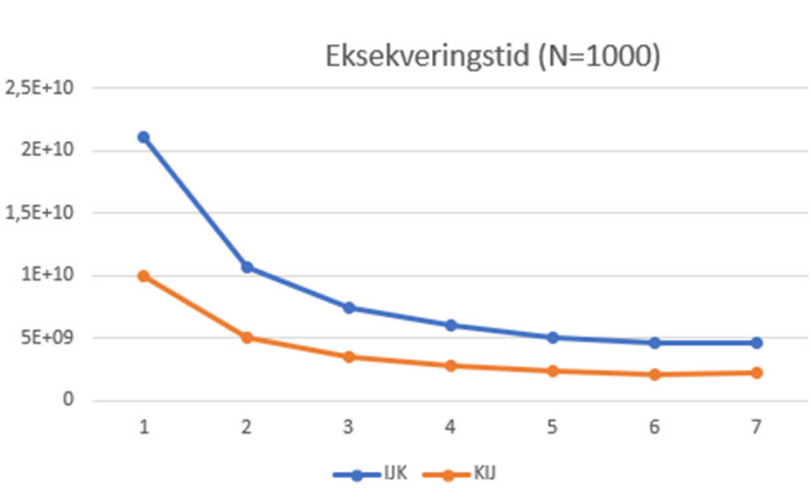
- Spaghetti-kode (call-back-hell): ”promises” til at beskrive ”naturlige” program ”sekvenser”
- Ikke egnet
  - til lange/tunge beregninger: skal manuelt opdeles
  - til parallelitet (workaround kræver specielle ”workers”)
- Tråde er en anden model!
  - Med preemption (time-slicing og I/O)
  - Kræver synkroniseringsmekanisme som locks+conditions/semaforer

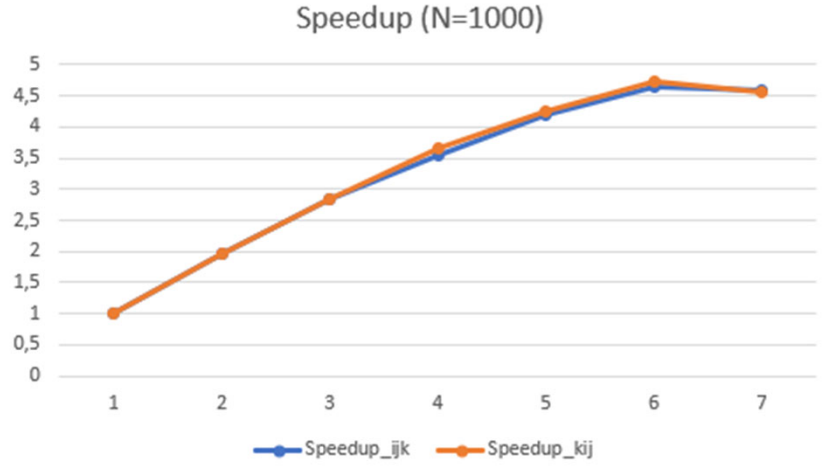
Opgaverne

- Hvornår har man gavn af at bruge flere tråde?
- Mulige udfald af program med race-condition?
- Hvordan beskytter vi kritiske regioner med låse?
  - Legetøjs bank-konti
  - Udvidelse af den parallelle approximative tæller

- Challenge 13:
  - Hvordan kan mm-multiplikation laves parallelt med tråde ?
  - og hvilken gevinst kan i opnå?




Eksekveringstid (N=1000)
2,5E+10
2E+10
1,5E+10
1E+10
5E+09
0
1
2
3
4
5
6
7
IJK
KIJ


Speedup (N=1000)
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
1
2
3
4
5
6
7
Speedup_ijk
Speedup_kij