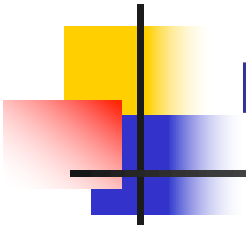


IN3030 L5, v2022 – More Synchronization

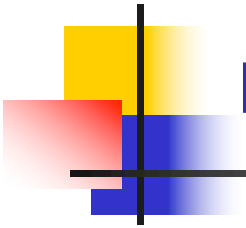
Eric Jul
Programming Technology Group
Department of Informatics
University of Oslo



Review L4

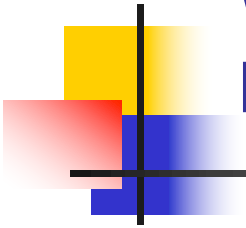
- Synchronization is HARD and ERROR PRONE!

I. Moore's Law
II. Speed of light
III. Why distribution
IV. Caches: WHAT and WHY
V. Matrice-multiplikasjon
VI. How to present your timing results
VII. Hva er PRAM modellen - og hvorfor er den ubrukelig for oss
VIII. Matrice-multiplikasjon



Plan L5

- Review of Synchronization Sketch
- Java's many ways to synchronize
- Synchronization explored via example: FinnMax
- Modellkode2-forslag for testing av parallell kode
- Finne primtall – divisjonsmetoden
- Finne primtall – Eratosthenes Sil (ES)



Which way shall we synchronize? Exploring possibilities using an example: FinnMax

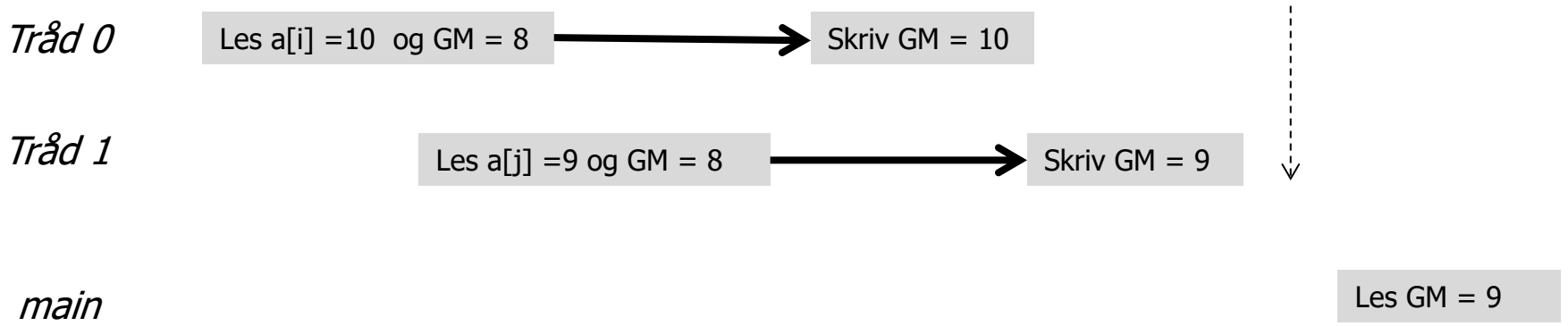
- Synchronization example: FinnMax

1) GAL – ingen synkronisering, koden i trådene:

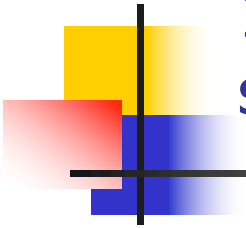
```
// Riktig oppdeling
ant= a.length/antTråder;   // antall elementer per tråd
start = ant*ind;
num = ant;
if (ind == antTråder-1) num = ant + (a.length % antTråder) ;
```

```
// GAL – IKKE synkronisert:
for (int i = 0; i< num; i++) {
      if (a[start+i] > globalMax) globalMax = a[start+i];
 }
```

- Mulig feilsituasjon, anta tråd 0 leser a[i] der GlobalMax ligger:



Hvor sannsynlig er dette? Skal vi ta sjansen ??



1) Nei – det er like raske og ALLTID riktige løsninger, og slike feil er farlige !

- Det er vanskelig å få denne gale måten til å feile, men med mye prøving og feiling (lure data) kan vi få den til å feile ca. ca. hver 100 000 gang.
- I Nets (tidl. Bankenes Betaligssentral) behandler de opp til 2 mill. transaksjoner hver dag.
- Vil vi lage programmer som kanskje feiler 20 ganger per dag?
- Hvor lette tror du slike programmer er å debugge (teste og rette) ?

2) Bruk av synchronized metode for alle elementene i a[]

```
 synchronized void updateGlobalMax1 (int val){
                        if (val > globalMax )  globalMax=val;
} // end updateGlobalMax1


.............
// i run-metoden i hver tråd:
  for (int i = 0; i< num; i++) {
            updateGlobalMax1(a[start+i]);
 }
```

- 100 % riktig svar alltid
- Det gjøres n kall på den synkroniserte metoden (n/k kall fra hver av de k trådene)
- synchronized tar mye tid !
- Elendig løsning: Regel 1: bruk ikke synchronized
- Regel 2: Hvis synchronized alikevel, så max 100 gange!

3) Bruk av ReentrantLock metode for alle elementene i a[]

```
void updateGlobalMax2 (int val){
            lock.lock();
            try{
                  if (val > globalMax )  globalMax=val;
            } finally {  lock.unlock(); }
} // end updateGlobalMax2
.....
// i run-metoden i hver tråd
for (int i = 0; i< num; i++) {
        updateGlobalMax2(a[start+i]);
}
```

- 100 % riktig svar alltid
- Det gjøres n kall på lock metoden (n/k kall fra hver av de k trådene)
- Lock er raskere enn synchronized, men tar fortsatt mye tid !
- Elendig løsning

4) Bruk av ReentrantLock metode hver gang nytt element er større

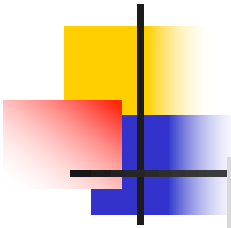
```
void updateGlobalMax2 (int val){
            lock.lock();
            try{
                  if (val > globalMax )  globalMax=val;
            } finally {  lock.unlock(); }
} // end updateGlobalMax2
.....
// i run-metoden i hver tråd
for (int i = 0; i< num; i++) {
        if (a[start+i] > globalMax ){
              updateGlobalMax2(a[start+i]);
 }}
```

- 100 % riktig svar alltid
- Det gjøres log n kall på lock metoden (logn/k kall fra hver av de k trådene)
- Lock er raskere enn synchronized, og logn kall er mye mindre enn n (eks. n= 1mill, log n =20) !
- Ikke helt bra løsning

5) lokal max (threadMax) i hver tråd, så oppdatering med lock - metode

```
// i run-metoden i hver tråd
for (int i = 0; i< num; i++) {
       if (a[start+i] > threadMax ) threadMax = a[start+i];
 }
  updateGlobalMax2(threadMax);
}
```

- 100 % riktig svar alltid
- Det gjøres k kall på lock-metoden (ett kall fra hver av de k trådene)
- Utmerket løsning



6) lokal max (threadMax) i hver tråd, så venting i ny CyclicBarrier (ventM5) , og tråd 0 ordner opp

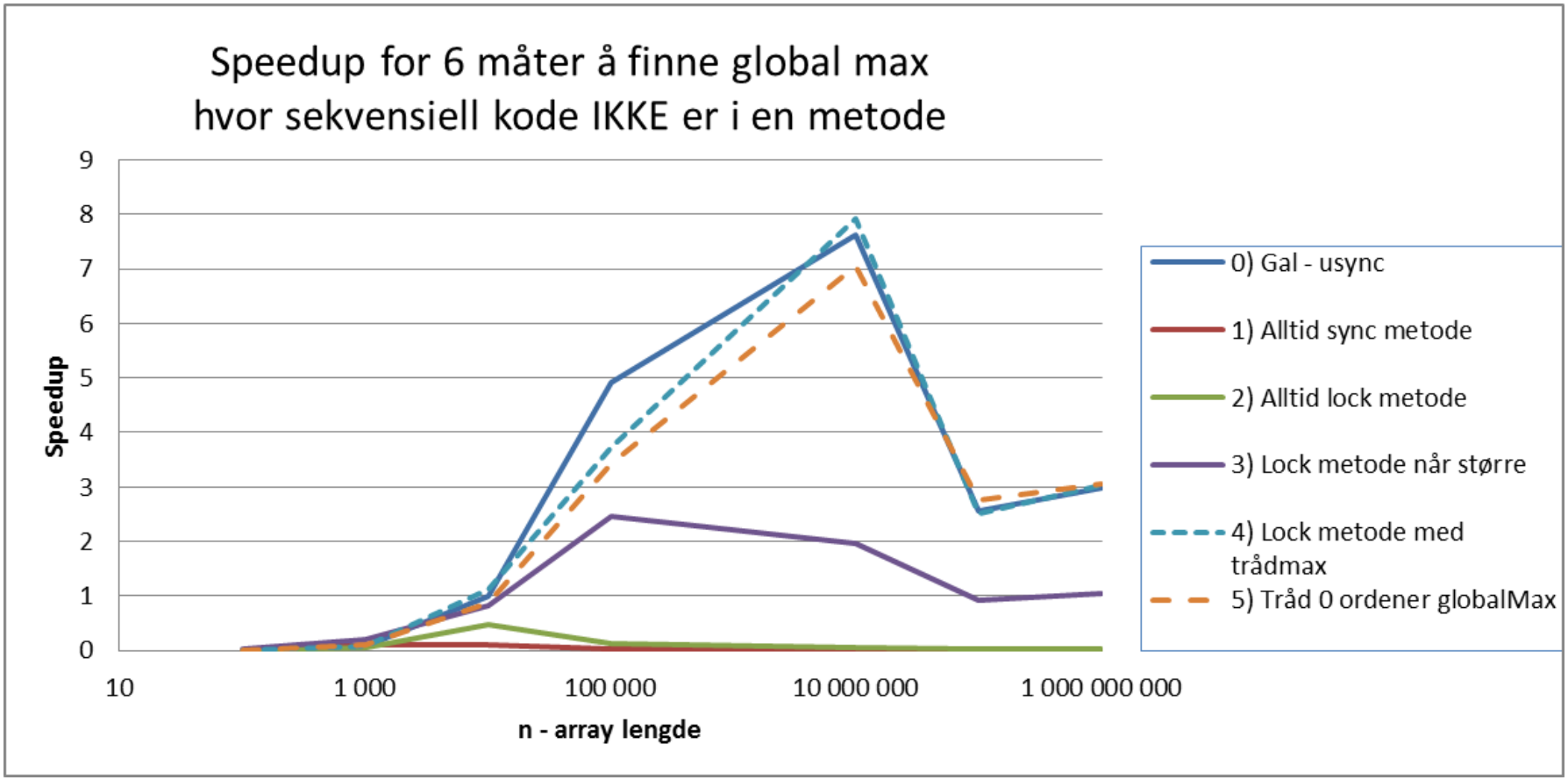
```
// i hver tråd i run-metoden
    for (int i = 0; i < num; i++) {
          if (a[start+i] > threadMax ) threadMax = a[start+i];
     }
 lokalMax[ind] = threadMax;

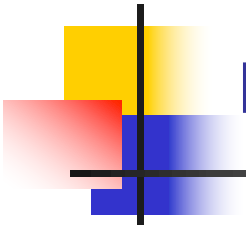
 try {  // wait on all other threads
       ventM5.await();
} catch (Exception e) {return;}

 if (ind == 0) {
     for (int j = 0; j < antTråder; j++){
           if (lokalMax[j] > globalMax) globalMax = lokalMax[j];
 } }
```

- Riktig og det gjøres k kall på en CylicBarrier(ett kall fra hver av de k trådene)
- Utmerket løsning

Speedup for n=100,1000,…,1 mrd med 8 tråder/kjerner





Resume of synchronizations

Different synchronizations for finding Max in an array:

1. No synch: WRONG
2. Use Java's `synchronized` keyword: SLOW
3. Using a ReentrantLock: BETTER BUT STILL SLOW
4. Using a Reentrant Lock only when necessary: BETTER
5. Postponing synchronization by using local max and then synching ONLY at the end: EXCELLENT
6. Using CyclicBarrier & letting Thread 0 find max of all the local max *without* synchronization: EXCELLENT